

Advances in Intelligent and Autonomous Aerospace Systems

EDITED BY

John Valasek

*Texas A&M University
College Station, Texas*



Volume 241

Progress in Astronautics and Aeronautics

Timothy C. Lieuwen, Interim Editor-in-Chief

*Georgia Institute of Technology
Atlanta, Georgia*

Published by
American Institute of Aeronautics and Astronautics, Inc.
1801 Alexander Bell Drive, Reston, VA 20191-4344



American Institute of Aeronautics and Astronautics, Inc., Reston, Virginia

1 2 3 4 5

Copyright © 2012 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. Reproduction or translation of any part of this work beyond that permitted by Sections 107 and 108 of the U.S. Copyright Law without the permission of the copyright owner is unlawful. The code following this statement indicates the copyright owner's consent that copies of articles in this volume may be made for personal or internal use, on condition that the copier pay the per-copy fee (\$2.50) plus the per-page fee (\$0.50) through the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, Massachusetts 01923. This consent does not extend to other kinds of copying, for which permission requests should be addressed to the publisher. Users should employ the following code when reporting copying from the volume to the Copyright Clearance Center:

978-1-60086-897-9/12 \$2.50 + .50

Data and information appearing in this book are for informational purposes only. AIAA is not responsible for any injury or damage resulting from use or reliance, nor does AIAA warrant that use or reliance will be free from privately owned rights.

ISBN 978-1-60086-897-9

PREFACE

Intelligent systems are entities that interact with their environment in such a way as to achieve their goals. Intelligent and autonomous agents or entities that handle decision and control functions are responsible for ensuring that systems perform properly and meet prescribed performance objectives. Consequently, the safe, reliable, and efficient operation of these agents or entities is essential in the domain of aerospace systems. Although no formal and widely accepted definition exists of what exactly makes an aerospace system intelligent and autonomous, some combination of the following characteristics is likely to be inherent:

1. Learning: having goals and sensing its environment it learns, for each system state or situation, which action permits it to achieve its goals
2. Reasoning: introspection/having a model of self/managing internal states
3. Deliberating/planning rather than simply reacting: continually acting, mentally and externally, and by acting reaches its objectives more often than pure chance indicates (normally much more often)
4. Adaptability: altering the state of the system such as its location in the environment or its internal state
5. Robustness to environment: generalizing appropriately to new situations and improving behavior given experience, including when the environment is nondeterministic, incompletely observable, and changing over time
6. Improving efficiency (over time and/or space): producing sequences of jointly optimized actions
7. Information compression: processing data to generate knowledge
8. Metabolization: consuming energy or resources and using it for its internal processes and in order to act

Research advances hold the promise to transform aerospace with systems that respond more quickly (e.g., autonomous collision avoidance, online path planning); work in dangerous or inaccessible environments (e.g., autonomous systems for space exploration); provide large-scale, distributed coordination (e.g., automated air traffic control, cooperative search and rescue); are highly efficient (e.g., morphing air vehicles, morphing space antennas); and augment human capabilities (e.g., next-generation glass cockpits, automated crew advisory systems). All of these capabilities will be realized by embedding computational intelligence, communication, control, and new mechanisms for sensing, actuation, and adaptation into air and space vehicles, propulsion systems, exploration systems, and vehicle management systems to name but a few.

This book seeks to provide both the aerospace researcher and the practicing aerospace engineer with an exposition on the latest innovative methods and approaches that focuses on intelligent and autonomous aerospace systems. The chapters are written by leading researchers in this field and include ideas,

directions, and recent results on current intelligent aerospace research issues with a focus on dynamics and control, systems engineering, and aerospace design. The content on uncertainties, modeling of large and highly nonlinear complex systems, robustness, and adaptivity are intended to be useful in the subsystem and the overall system level design and analysis of various aerospace vehicles. A broad spectrum of methods and approaches is presented, including bio-inspiration; fuzzy logic; genetic algorithms; Q-learning; Markov decision processes; approximate dynamic programming; artificial neural networks; probabilistic maps; multi-agent systems; and Kalman, particle, and confidence filtering.

Although an original work, this book has its roots in a similar book written by Ewald Heer and Henry Lum from 1989 titled *Machine Intelligence and Autonomy for Aerospace Systems* and published by AIAA in the Progress in Aeronautics and Astronautics Series. It is enlightening to review the scope and content of that work to see how the field has progressed in the last quarter-century. This work focused exclusively on spacecraft that were currently anticipated from contemporary requirements and applications on future space missions. It was envisioned that these would present building blocks for further development and advances. Machine intelligence and human-machine interaction aspects were emphasized as a means to increase the technical feasibility and economic feasibility of spacecraft systems. As defined in that work, these aspects described systems that accept human commands, reason, and perform manipulative tasks.

It is interesting that expert systems and knowledge-based systems were featured so prominently in this earlier work because they did not see nearly the widespread use that was envisioned for them. On the other hand, information acquisition for single and multisensor systems was identified as an important emerging area, and this is certainly true today and for the foreseeable future.

A rough indicator of the level of maturation of the intelligent systems of a quarter-century ago can be inferred from the fact that only one of the 12 chapters of the book featured a system or application that used closed-loop intelligent or autonomous control. The application was the control of a deformable spacecraft system. By contrast, seven of the 12 chapters of this book feature closed-loop intelligent or autonomous control. It is clear that the field has matured to a level in which closed-loop control is increasingly pervasive. Whether or not this trend continues remains to be seen.

This book is organized broadly into three parts. Although many different criteria could have been used to group the chapters, applying a general form of the technique of classifiers eventually produced the good but by no means unique organization used. In spite of this, many of the chapters have distinct technical elements that could have easily justified their inclusion in a different part. This serves to remind one of the interdisciplinary nature of intelligent systems. Part I is titled Intelligent Flight Control and presents work on various control techniques applied to flapping micro air vehicles, unmanned helicopters, and generic transport aircraft. Part II is titled Intelligent Propulsion and Health Management and features architectures for integrated health management and health

monitoring, as well as two chapters on propulsion control. Part III is titled Intelligent Planning and Multi-Agent Systems and addresses a variety of methods to support and conduct missions ranging from autonomous soaring to cooperative teams of UAVs to space exploration missions to air vehicle search and target tracking.

It is my sincere hope that the reader finds the book as useful and rewarding as it has been writing it.

John Valasek

College Station, Texas

June 2012

TABLE OF CONTENTS

Acknowledgment	xiii
-----------------------------	-------------

Preface	xv
----------------------	-----------

PART I INTELLIGENT FLIGHT CONTROL

Chapter 1 Towards Bio-Inspired Robotic Aircraft: Control Experiments on Flapping and Gliding Flight	1
--	----------

Michael Dorothy, Aditya A. Paranjape, P. Daniel Kuang and Soon-Jo Chung,
University of Illinois at Urbana–Champaign, Urbana, Illinois

I. Introduction	1
II. Biologically Inspired CPG Control Basics	4
III. Kinematics and Unsteady Aerodynamics	7
IV. CPG-Based Control Results of RoboBat	11
V. Flight Mechanics of MAV with Articulated Wings	14
VI. Control Law Design	16
VII. Experiments on the Elements of Perching	21
VIII. Conclusion	28
Acknowledgments	29
References	29

Chapter 2 Neural Network-Based Optimal Control of an Unmanned Helicopter	33
---	-----------

David Nodland and H. Zargarzadeh, *Missouri University of Science and Technology, Rolla, Missouri*; Arpita Ghosh, *National Metallurgical Laboratory, Jamshedpur, India*;
and S. Jagannathan, *Missouri University of Science and Technology, Rolla, Missouri*

I. Introduction	33
II. Helicopter Dynamic Model	36
III. Nonlinear Optimal Regulation and Tracking of the Helicopter	39
IV. Simulation Results	52
V. Conclusion	56
Acknowledgments	56
References	56

Chapter 3 Intelligent Constrained Optimal Control of Aerospace Vehicles with Model Uncertainties 59

Jie Ding and S. N. Balakrishnan, *Missouri University of Science and Technology, Rolla, Missouri*

Nomenclature	59
I. Introduction	60
II. Approximate Dynamic Programming	63
III. J-SNAC Synthesis with Nonquadratic Cost	65
IV. Dynamic Reoptimization of J-SNAC Controller	67
V. Tracking Problem with Input Constraint	69
VI. Numerical Results	70
VII. Conclusion	84
Appendix	85
Acknowledgments	87
References	87

Chapter 4 Modified Reference Model MRAC (M-MRAC): An Application to a Generic Transport Aircraft 91

Vahram Stepanyan and Kalmanje Krishnakumar, *NASA Ames Research Center, Moffett Field, California*

I. Introduction	91
II. Preliminaries	93
III. Reference Model Modification	98
IV. Asymptotic Properties of M-MRAC	100
V. Transient Properties of M-MRAC	101
VI. Disturbance Rejection	108
VII. Scalar Example	111
VIII. Aerospace Application	117
IX. Conclusions	125
References	126

Chapter 5 \mathcal{L}_1 Adaptive Control in Flight 129

Enric Xargay and Naira Hovakimyan, *University of Illinois at Urbana–Champaign, Urbana, Illinois*; Vladimir Dobrokhodov and Isaac Kaminer, *Naval Postgraduate School, Monterey, California*; Chengyu Cao, *University of Connecticut, Storrs, Connecticut*; and Irene M. Gregory, *NASA Langley Research Center, Hampton, Virginia*

I. Fast Adaptation: The Key to Safe Flight	129
II. \mathcal{L}_1 Adaptive Control for the NPS Autonomous UAV	132
III. \mathcal{L}_1 Adaptive Control for the NASA AirSTAR Flight-Test Vehicle	140

IV. Concluding Remarks and Future Research Directions 168

Acknowledgments 169

References 169

PART II INTELLIGENT PROPULSION AND HEALTH MANAGEMENT

Chapter 6 Integrated Systems Health Management for Intelligent Systems 173

Fernando Figueroa, *NASA Stennis Space Center, Mississippi* and Kevin Melcher, *NASA John H. Glenn Research Center at Lewis Field, Cleveland, Ohio*

Acronyms 173

I. Introduction 174

II. iISHM Capability Development 175

III. iISHM in Systems Design, Integration, and Engineering 189

IV. Intelligent Control for iISHM-Enabled Systems 191

V. Opportunities and Need for Advances in Verification and Validation 192

VI. Implementation Example: Rocket-Engine Test Facility and Test Article 192

VII. Conclusion 197

Acknowledgments 198

References 198

Chapter 7 Intelligent Propulsion Control and Health Management 201

Sanjay Garg, *NASA John H. Glenn Research Center at Lewis Field, Cleveland, Ohio*

I. Introduction 201

II. Turbofan Engine Overview 202

III. State of the Art of Engine Control 204

IV. Some Retrofit Intelligent Engine Control Concepts 209

V. Model-Based Control and Diagnostics 216

VI. Conclusion 226

References 227

Chapter 8 Genetic Fuzzy Controller for a Gas-Turbine Fuel System 229

Andrew Vick and Kelly Cohen, *University of Cincinnati, Cincinnati, Ohio*

I. Introduction to Genetic Fuzzy Systems 229

II. Gas-Turbine Fuel System	246
III. Genetic Fuzzy Gas-Turbine Fuel System	248
IV. Conclusions	269
References	271

PART III INTELLIGENT PLANNING AND MULTI-AGENT SYSTEMS

Chapter 9 Multiresolution State-Space Discretization Method for Q-Learning for One or More Regions of Interest 273

Amanda Lampton and John Valasek, *Texas A&M University, College Station, Texas*

I. Introduction	273
II. Reinforcement Learning	276
III. Learning on a Two- and N-Dimensional Continuous Domain	281
IV. Multiresolution State-Space Discretization (AAG) for N Dimensions	283
V. Multiple Goal Regions	285
VI. Policy Comparison	286
VII. Benchmark Dynamic System Example — Inverted Pendulum	288
VIII. Autonomous Soaring Updraft/Thermal Example	298
IX. Conclusions	306
Acknowledgments	307
References	307

Chapter 10 Motion Planning Under Uncertainty 309

Ali-Akbar Agha-Mohammadi, *Texas A & M University, College Station, Texas*;
Sandip Kumar, *Mathworks, Natick, Massachusetts*; and Suman Chakravorty,
Texas A & M University, College Station, Texas

I. Introduction	309
II. Mathematical Formulation for Decision Making Under Uncertainty	318
III. Generic Framework for Sampling-Based Feedback Motion Planners	322
IV. Sampling-Based Feedback Motion Planners with Process Uncertainty and Stochastic Maps	330
V. Sampling-Based Feedback Motion Planners with Motion Uncertainty and Imperfect State Information	345
Appendix A: Proof of Lemma 4	374
Appendix B: Proof of Corollary 1	376
Appendix C: Proof of Proposition 1	376

Appendix D: Proof of Theorem 1	380
References	381

Chapter 11 Protocol Utilization in Intelligent Systems to Facilitate Exploration Missions 387

M. Lyell and W. Drozd, <i>Intelligent Automation, Inc., Rockville, Maryland</i> ; A. Grinberg Webb, <i>Montgomery College, Rockville, Maryland</i> ; and J. Nanda and W. Chen, <i>Intelligent Automation, Inc., Rockville, Maryland</i>	
I. Introduction	387
II. Mission Criteria and Behavior Norms: The Impact on Framework Design	390
III. Technology Fundamentals: Software Agent Paradigm and Protocols	394
IV. Multi-Agent-Based Subsystems and Astronaut Interactions and Oversight: Behavior and Protocol Development	395
V. Robot Agent	407
VI. Autonomous Robot Teams and Astronaut Oversight: Protocol Development	411
VII. Protocol Utilization by the Deployed Robot Team	428
VIII. Discussion and Conclusions	438
Acknowledgment	441
References	441

Chapter 12 Framework for User-Guided Search and Adaptive Target Tracking via Cooperative UAVs 445

R. A. Cortez, D. Tolic and I. Palunko, <i>University of New Mexico, Albuquerque, New Mexico</i> ; N. Eskandari, <i>University of British Columbia, Vancouver, British Columbia, Canada</i> ; and M. Oishi, R. Fierro and I. Wood, <i>University of New Mexico, Albuquerque, New Mexico</i>	
I. Introduction	445
II. Problem Formulation	447
III. UAV Coordination	449
IV. Quadrotor Baseline Controller Design	456
V. Simulations and Experiments	460
VI. Conclusions and Future Work	465
References	467

Index	469
------------------------	------------

Supporting Materials	491
---------------------------------------	------------

Towards Bio-Inspired Robotic Aircraft: Control Experiments on Flapping and Gliding Flight

Michael Dorothy,^{*} Aditya A. Paranjape,[†] P. Daniel Kuang,[‡]
and Soon-Jo Chung[§]

University of Illinois at Urbana–Champaign, Urbana, Illinois 61801

I. INTRODUCTION

THERE is a growing interest in the aerospace community in the development of robotic micro aerial vehicles (MAVs) to learn and mimic avian flight. MAVs fly in low-Reynolds-number regimes of 10^3 to 10^5 , which corresponds to that of small birds or bats [1]. MAVs with wings equipped with multiple degrees of freedom such as flapping, wing twist, and sweep provide greater payload capability than insect-like MAVs and greater maneuverability than conventional fixed-wing aircraft. These MAVs can be used for intelligence gathering, surveillance, and reconnaissance missions in tightly constrained spaces such as forests and urban areas. Advances in actuators and control systems have led to development and analysis of articulated and flapping MAVs inspired by animals [2–5]. Birds and bats achieve remarkable stability and perform agile maneuvers using their wings very effectively [2]. One of the goals of reverse-engineering animal flight is to learn more about the various aspects of avian flight such as stability, maneuverability, and control from the dynamics of MAV.

From a controls standpoint, there are three major flight regimes or modes in animals: high-frequency flapping, low-frequency flapping, and gliding. Insects reside almost entirely in the high-frequency domain, utilizing small musculature to produce passive pitching dynamics over the course of several wingbeats [6]. For these situations, we can utilize averaging theorems to greatly aid control design around trim states such as hover condition [3, 7, 8]. Much bat and bird flight is low frequency enough that averaging theorems do not apply in practice and intra-wingbeat effects are significant [9]. Additionally, steady-state equilibria (or approximations thereof) do not exist, which distinguishes this regime from the other two. Dietl and Leonard give examples of an analysis of this regime by

^{*}Doctoral Student, Department of Aerospace Engineering; dorothy1@illinois.edu. Student Member AIAA.

[†]Doctoral Student, Department of Aerospace Engineering; adityaparanjape@gmail.com. Student Member AIAA.

[‡]Master's Student, Department of Aerospace Engineering; pdkuang2@gmail.com. Student Member AIAA.

[§]Assistant Professor, Department of Aerospace Engineering; sjchung@illinois.edu. Senior Member AIAA.

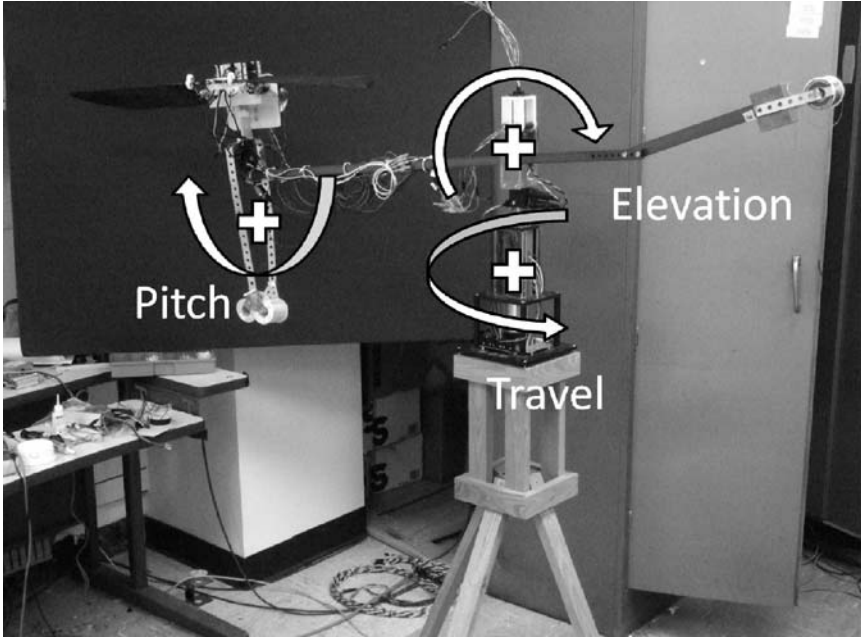


FIG. 1 RoboBat testbed.

Floquet theory [10, 11]. With the exception of one isolated case in [11] (a steep climb with very small negative eigenvalues), all such Floquet investigations have shown instability. Biologically, bat flight looks more like walking locomotion in other mammals, where many joints must be coordinated in a rhythmic fashion to produce motion. Bats have anatomical similarities with other mammals. Although the authors are not aware of tests specifically on bats, it makes sense that they would coordinate their joints in a similar fashion to many other mammals: with central pattern generators (CPGs) [12, 13].

Previous robotic flapping flyers and their control design consider one or two degrees of freedom in the wings [3, 4, 7, 14]. However, even insects like the dragonfly (*Anax parthenope*) are reported to have complex three-dimensional movements by actively controlling flapping and twisting of four independent wings [15]. Furthermore, prior studies in flapping flight [1, 4, 15–19] assumed a very simple sinusoidal function for each joint to generate flapping oscillations, without deliberating on how multiple limbs (or their nervous systems) are connected and actuated to follow such a time-varying reference trajectory.

To utilize the knowledge gained from CPGs in biological fliers, we have built a robotic bat, shown on a pendulum testbed in Fig. 1 with dimensionality far lower than the animals. These experiments are an early step toward strict modeling of biological fliers, but are more helpful for design of an artificial flapping flier. We exploit

the dimensional reduction made possible by simple CPG rules to make control design and aerodynamic testing feasible. In time, we expect to gain insight into the stability and agility of animal flight, though we hope to encounter engineered solutions that are even more efficient than their biological counterparts.

Moderately large birds and bats often spend their time in either a low-frequency flapping mode or a gliding mode. The proposed CPG-based controller can switch smoothly to the gliding mode by changing a bifurcation parameter. The gliding mode is not unlike that explored in the traditional flight mechanics literature. However, fully articulated wings inherent in flapping flight create additional control possibilities and concerns. A gliding mode can be used for soaring and to shed energy in preparation for a perching maneuver. Perching can be described as a high angle-of-attack pull-up with high lift and a large drag. The large lift and drag forces cause the MAV to climb and lose speed significantly. A planted landing can be achieved in the process.

Birds successfully perch on a variety of structures such as building ledges, power lines, cliff side, and tree branches. Such perching capability in MAVs can significantly reduce the landing distance. However, perching requires the ability to maintain trajectory very accurately. Furthermore, a typical perching maneuver would not last more than a few seconds. Because of its duration and highly unsteady flight profile, perching is an important agility metric for MAVs. The unsteady flight profile makes control design for perching a challenging problem.

The aerodynamics of perching has been explored for conventional, fixed-wing aircraft by Crowther [20] who showed that perching could be performed with essentially a simple pitch-up maneuver and used genetic algorithm to optimize the maneuver. Wickenheiser and Garcia demonstrated perching maneuver with controlled wing twist and variable tail incidence [21, 22]. Roberts *et al.* [23] showed that controllability is not lost even under high-alpha, low speed conditions. One of the most outstanding experimental demonstrations of a perching maneuver was reported by Cory and Tedrake [24] where they obtained reliable estimates of the open-loop dynamics and used them to perform a maneuver optimized to minimize the error in the final position. In contrast with the aforementioned work, we consider a completely different mechanism (wing dihedral) to control the flight path angle as well as lateral-directional dynamics during perching. The lateral-directional control, in particular, is often neglected in the literature on robotic perching. Applications including flexible wings, especially for flapping vehicles, should consider root or tip-based boundary control of partial differential equations [25, 26].

The broader objective of the work described in this paper is to aid the design and analysis of a robotic aircraft capable of agile flight in the flapping as well as gliding phases. The immediate objective of this paper is two-fold: 1) to demonstrate the importance of phase differences in stability of low-frequency flapping flight and 2) to demonstrate the essential elements of a perching maneuver in gliding MAVs, which use a combination of wing dihedral and elevator for control. A CPG-like control design is used for the flapping phase and for transitioning to the gliding phase for which a more conventional controller is employed. The rest of

the paper is organized as follows. In what may be considered part one, we review our experimentation with RoboBat, a flapping testbed. We introduce CPG control in Sec. II, provide a full dynamic model of the testbed in Sec. III, and show stability and control of longitudinal modes in Sec. IV. In what may be considered part two, we review our experiments with an articulated wing gliding testbed. We summarize the mechanics of known articulated wing flight in Sec. V, develop the control law in Sec. VI, and present closed-loop results in Sec. VII.

II. BIOLOGICALLY INSPIRED CPG CONTROL BASICS

The flapping motion in bats combines many different joints and muscles to create rhythmic motions that produce the aerodynamic forces required to sustain flight. With the current state of the actuator/structure technology, we are unable to mimic the strength and flexibility of the distributed structure of bones, joints, and ligaments. More specifically, we are unable to produce muscle-strength actuation of the entire system using mechanical actuators. For this purpose, we have settled on starting our investigation with three main modes of motion: flapping, twisting, and lead-lag (sometimes referred to as plunging, pitching, and sweeping). These three motions (shown in Fig. 2) have the strongest connection to traditional flight models and were the first to be discussed in observations of animal flight [15]. Azuma also notices that the phase differences between the motions are extremely important [15]. Next, we consider how to use CPGs for precise control of phase differences between the three motions.

A. CENTRAL PATTERN GENERATORS

Many creatures produce their motion by synchronizing periodic motions of limbs, such as running, swimming, or flapping. They do this by coupling biological oscillators and synchronizing their outputs. Biological oscillators rely on short timescale (ms) neuron dynamics including spike-bursting, spike frequency adaptation, and postinhibitory rebound. Herrero-Carrón et al. [27] designed a control law for modular robots by approximating short timescale neuron dynamics. Because there is such a short timescale required for integration, the neuron dynamics

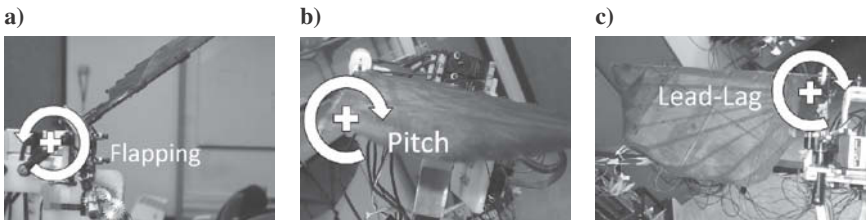


FIG. 2 Main wing motions used by RoboBat: a) RoboBat flapping motion ϕ , b) RoboBat pitching motion θ , and c) RoboBat lead-lag motion ψ .

were integrated offline. We are unlikely to be able to perform such strict mimicry in an online controller as we add additional neurons for feedback, active control of phase differences, or gait transitions.

To make online control more feasible, we can emulate these biological oscillators by using coupled nonlinear limit-cycle oscillators. A limit-cycle oscillator converges to a stable trajectory, which is called the limit cycle. Because of this convergence, the oscillator will quickly forget sporadic disturbances and converge back to the stable limit cycle. If the oscillator itself is a smooth vector field, we can smoothly transition between desired trajectories without abrupt changes being required in the motor output.

This type of control design allows the complicated synchronization and trajectory computations to be performed according to simple rules that provide the desired oscillatory behavior. If we can then influence body motion by simply tuning top-level inputs into the CPG network, we can achieve the dimension reduction that allows the control problem to be computationally feasible. Vertebrate brains send top-level chemical signals that modulate, start, and stop CPG behavior, but do not micromanage the joint trajectories [28].

Following our previous work [5], we use the following limit-cycle model called the Hopf oscillator, named after the supercritical Hopf bifurcation model with $\sigma = 1$:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}; \rho; \sigma) + \mathbf{u}(t) \quad \text{with } \mathbf{x} = (u - a, v)^T$$

and

$$\mathbf{f}(\mathbf{x}; \rho; \sigma) = \begin{bmatrix} -\lambda \left(\frac{(u - a)^2 + v^2}{\rho^2} - \sigma \right) & -\omega(t) \\ \omega(t) & -\lambda \left(\frac{(u - a)^2 + v^2}{\rho^2} - \sigma \right) \end{bmatrix} \mathbf{x} \quad (1)$$

where the $\lambda > 0$ denotes the convergence rate to the symmetric limit circle of the radius $\rho > 0$ and $\mathbf{u}(t)$ is an external or coupling input. The bifurcation parameter σ can change to a negative number {e.g., -1 such that $[(u - a)^2 + v^2/\rho^2 + 1]}$. This would change the stable limit-cycle dynamics to the dynamics with a globally stable equilibrium point at the bias a [29]. Such a change can be used to turn the flapping oscillatory motion to the gliding mode, as was seen in [5]. This could be used to mimic many animal flight strategies. Often, bats and birds will transition to gliding in order to shed energy well before an actual perching maneuver is performed. Later sections of this paper discuss terminal perching control, assumed to be in a gliding phase. In [5], it was shown that coupled networks of Hopf oscillators on balanced graphs exhibit smooth exponentially stable behavior in both oscillatory mode and fixed point mode.

The possibly time-varying parameter $\omega(t) > 0$ determines the oscillation frequency of the limit cycle. A time-varying $a(t)$ sets the bias to the limit cycle such that it converges to $u(t) = \rho \cos(\omega t + \delta) + a$ and $v(t) = \rho \sin(\omega t + \delta)$ on

a circle. The output variable to generate the desired oscillatory motion of each joint is the first state u from the Hopf oscillator model in Eq. (1).

Phase synchronization means an exact match of the scaled amplitude or the frequency in this paper. Hence, phase synchronization permits different actuators to oscillate at the same frequency but with a prescribed phase lag. In essence, each CPG dynamic model in Eq. (1) is responsible for generating the limiting oscillatory behavior of a corresponding joint, and the diffusive coupling among CPGs reinforces phase synchronization. For example, the flapping angle has roughly a 90-deg phase difference with the pitching joint to maintain the positive angle of attack (see the actual data from birds in [15]). The oscillators are connected through diffusive couplings, and the i th Hopf oscillator can be rewritten with a diffusive coupling with the phase-rotated neighbor.

$$\dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i; \rho_i) - k \sum_{j \in \mathcal{N}_i}^{m_i} \left[\mathbf{x}_i - \frac{\rho_i}{\rho_j} \mathbf{R}(\Delta_{ij}) \mathbf{x}_j \right] \quad (2)$$

where the Hopf oscillator dynamics $\mathbf{f}(\mathbf{x}_i; \rho_i)$ with $\sigma = 1$ is defined in Eq. (1), \mathcal{N}_i denotes the set that contains only the local neighbors of the i th Hopf oscillator, and m_i is the number of the neighbors. The 2×2 matrix $\mathbf{R}(\Delta_{ij})$ is a two-dimensional rotational transformation of the phase difference Δ_{ij} between the i th and j th oscillators. The positive (or negative) Δ_{ij} indicates how much phase the i th member leads (or lags) from the j th member and $\Delta_{ij} = -\Delta_{ji}$. The positive scalar k denotes the coupling gain.

Numerous configurations are possible as long as they are on balanced graphs [30], and we can choose either a bidirectional or a unidirectional coupling between the oscillators. The numbers next to the arrows in Fig. 3 indicate the phase shift Δ_{ij} ; hence, $\Delta_{ij} > 0$ indicates how much phase the i th member leads. Figure 3a shows the choice of coupling used in this paper, where ϕ_w , θ_w , and ψ_w are flapping angle, twist angle, and lead-lag angle, respectively. Subscripts L and R refer to left and right wings. The graph is balanced. Further, all the phase shifts (Δ_{ij}) along one cycle add up to a modulo of 2π . The proof of stability of Eq. (2) is given in [5]. As mentioned in [31], this type of oscillator generalizes other types of waveform control such as the split cycle [8].

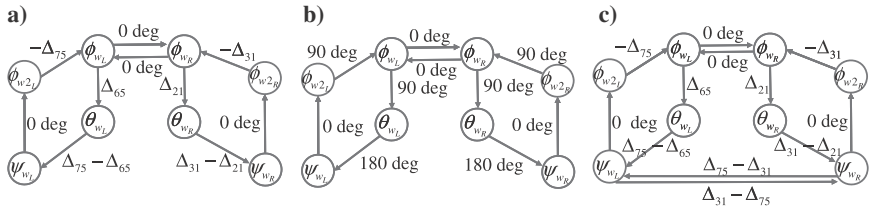


FIG. 3 Example graph configurations of the coupled Hopf oscillators on balanced graphs: a) configuration A, b) symmetric configuration A, and c) configuration B.

B. CONTROL DESIGN FROM PHYSICAL INTUITION AND BIOLOGICAL OBSERVATION

Our previous work [5] provided a simple intuition for phase-difference control of longitudinal motion in flapping flight. With a zero-bias lead-lag and a center of gravity coinciding with the stroke plane, a phase difference of 270 deg between the flapping CPG and the lead-lag CPG gives Azuma's [15] elliptical model of flapping: negative lead-lag on downstroke and positive lead-lag on upstroke. The simplest analysis combines a maximum force with the most negative lead-lag at the middle of the downstroke to predict a pitch-down moment on the body. Alternatively, if we see the phase difference to 90 deg, we see the maximum force coinciding with the maximum positive lead-lag at the middle of the downstroke, predicting a pitch-up moment. This intuition has been confirmed in numerical simulations and open-loop experiments on the RoboBat [5, 32, 33]. In Sec. IV, we attempt to exploit this intuition for closed-loop, low dimensional control of the RoboBat.

III. KINEMATICS AND UNSTEADY AERODYNAMICS

We present the dynamic model of the current RobotBat, which is not intended to be a free-flying platform. The detailed dynamic models of free-flying ornithopters can be found in [2, 5]. In addition, [34] includes wing flexibility. It is intended as a testbed for CPG control designs, experimental confirmation of unsteady aerodynamics, and experimental determination of optimal wing motions. The weight and power requirements have not been optimized for free flight. To test longitudinal control strategies, it has been attached to a Quanser pendulum platform, which provides encoder feedback signals that we can use for control. Figure 1 shows the three degrees of freedom: travel, elevation, and pitch (λ , ϵ , θ). For more information on the testbed, see [33].

Of note is the fact that the pitch rotation point is not near the center of gravity of the bat. To make experimentation feasible, we have affixed a counterweight on the pitch arm. By moving this counterweight or changing its mass, we can alter the natural stability of the pitch motion. One consequence of this scheme is that the pitch motion has an artificially high moment of inertia. Therefore, we expect that our moment-producing control schemes for a tailless vehicle will have even more effectiveness in a free flier. To move toward computations of actual forces and moments generated, we desire dynamic modeling of the pendulum setup and the unsteady aerodynamics. If we define our generalized coordinates to be $[q_1, q_2, q_3] = [\epsilon, \theta, \lambda]$, then using Lagrange's equations, $(d/dt)[\partial L(q, \dot{q})/\partial \dot{q}] - [\partial L(q, \dot{q})/\partial q] = F$ and algebraic manipulations, we can transform the equations of motion (EOM) to standard robot form [35].

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \quad (3)$$

The forces and moments on the right-hand side can be found as a function of wing kinematics and an aerodynamic model. The generalized forces remain

intact through the transformation to robot form, that is, $F = \tau$, and are computed to be

$$\tau = \begin{Bmatrix} \ell_a [\mathbf{L} \cos(q_2) + \mathbf{T} \sin(q_2)] \\ \mathbf{M} - \ell_b \mathbf{T} \\ \ell_a [\mathbf{T} \cos(q_2) - \mathbf{L} \sin(q_2)] \end{Bmatrix} \quad (4)$$

where \mathbf{L} , \mathbf{T} , and \mathbf{M} are found later in the aerodynamic model. This formulation can then be applied to a free-flying MAV. Here, we present a refinement on the aerodynamic model of [5].

The pendulum rig consists of the following:

1. A solid bar with mass M_p is hinged at its center of gravity such that it can spin about the vertical axis (angle given by λ , positive counterclockwise) and rotate upwards and downwards in the vertical plane (angle denoted by ϵ , positive downwards).
2. A compound pendulum is mounted on one end of the bar consisting of two point masses: the robotic bat itself modelled as a point mass m_b and a variable mass m . The compound pendulum is free to swing in the plane normal to the bar, with the swing angle given by θ .
3. A counterweight m_w is located at the opposite end of the bar as the bat.

Three frames of reference can be defined for this system, given an inertial frame of reference I fixed to the Earth:

1. A frame B is fixed to the compound pendulum with its origin at the suspension point. The frame B is parallel to the aircraft body-axis frame centered at the aircraft c.g.
2. A frame P exists with its origin at the bar's hinge point such that under nominal conditions the axes of P and B are parallel to each other.
3. A frame S is constructed locally at every wing station for calculation of the local wind velocity and the aerodynamic forces and moments.

The frame I is first rotated about the z axis by an angle λ , followed by a rotation about the x axis by ϵ to coincide with the P frame. Therefore, the following rotation matrix is obtained to transform the components of a vector from I to P :

$$\begin{aligned} R_{PI} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \epsilon & -\sin \epsilon \\ 0 & \sin \epsilon & \cos \epsilon \end{bmatrix} \begin{bmatrix} \cos \lambda & -\sin \lambda & 0 \\ \sin \lambda & \cos \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos \lambda & -\sin \lambda & 0 \\ \cos \epsilon \sin \lambda & \cos \epsilon \cos \lambda & -\sin \epsilon \\ \sin \epsilon \sin \lambda & \sin \epsilon \cos \lambda & \cos \epsilon \end{bmatrix} \end{aligned}$$

The frame P is rotated about the y axis to obtain frame B :

$$R_{BP} = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \quad (5)$$

The rotation matrix from B to S is obtained by performing the standard 3 – 2 – 1 rotations through angles ψ , θ_w , and $-\phi$, respectively:

$$R_{SB} = \begin{bmatrix} \cos \phi \cos \psi & \sin \psi \cos \phi & \sin \phi \\ -\sin \phi \sin \theta_w \cos \psi - \sin \psi \cos \theta_w & -\sin \phi \sin \psi \sin \theta_w + \cos \theta_w \cos \psi & \sin \theta_w \cos \phi \\ -\sin \phi \cos \theta_w \cos \psi - \sin \psi \sin \theta_w & -\sin \phi \sin \psi \cos \theta_w + \sin \theta_w \cos \psi & \cos \theta_w \cos \phi \end{bmatrix} \quad (6)$$

The matrix $R_{SP} = R_{SB} R_{BP}$ transforms the coordinates of a vector from the P frame to S frame and will be useful while calculating velocities and forces.

Let ω_P denote the angular velocity of the horizontal rod in the P frame, that is, $\omega_P = [\dot{\epsilon} \ 0 \ -\dot{\lambda}]^T$. Let $\omega_B = [0 \ \dot{\theta} \ 0]^T$ denote the angular velocity of the compound pendulum about O_B . Finally, let ω denote the angular velocity of a wing with components in the B frame (a similar expression can be derived for the other wing). It follows that

$$\omega = \begin{bmatrix} -\dot{\phi} - \dot{\psi} \sin \theta_w \\ \dot{\theta}_w \cos \phi - \dot{\psi} \cos \theta_w \sin \phi \\ \dot{\theta}_w \sin \phi + \dot{\psi} \cos \theta_w \cos \phi \end{bmatrix} \quad (7)$$

Let $Y = [0 \ l_a \ 0]^T$ denote the position vector from O_P to O_B . Let $z = [0 \ 0 \ -l_b]^T$ denote the position vector from O_P to the bat c.g. and $y(y) = [0 \ y \ 0]^T$ denote the vector from bat c.g. to a wing station with components in the S frame. Then, the local wind velocity at a wing station is given by

$$\begin{aligned} V = & \underbrace{R_{SP}S(\omega_P)Y + R_{SB}S(\omega_B + R_{BP}\omega_P)z}_{\text{equivalent to flight speed } V_\infty} \\ & + S[R_{SB}(\omega + \omega_B) + R_{SP}\omega_P]y \end{aligned} \quad (8)$$

The local angle of attack is given by

$$\alpha = \tan^{-1} \left(\frac{V_3}{V_1} \right) \quad (9)$$

where V_i is the i th element of V .

The local acceleration is assumed to arise entirely from flapping. Because the model developed in this paper is intended to be as generic as possible, the model proposed by Goman and Khrabrov [36] is presented as a candidate model for computing the lift and the quarter-chord moment while drag is estimated assuming the classic drag polar. In the authors' estimate, Goman and Khrabrov's model offers at least two advantages over the existing models (e.g., Theodorsen or Peters

[37]). First, the model is cast in the form of a single ordinary differential equation (ODE) and two algebraic equations, one each for life and the quarter-chord pitching moment. The state variable for the ODE corresponds, physically, to the chordwise location of flow separation on the airfoil. Therefore, the model is quite easy to implement as part of a numerical routine. Second, the model is inherently non-linear and applicable to poststall conditions.

The following equation describes the movement of the separation point for unsteady flow conditions

$$\tau_1 \dot{v} + v = v_0(\alpha - \tau_2 \dot{\alpha}) \quad (10)$$

where τ_1 is the relaxation time constant, τ_2 captures the time delay effects due to the flow, while v_0 is an expression for the nominal position of the separation point. These three parameters need to be identified experimentally or using computational fluid dynamics (CFD) for the particular airfoil under consideration. The coefficients of lift and quarter-chord moment are then given by

$$\begin{aligned} C_l^* &= \frac{\pi}{2} \sin[\alpha(1 + v + 2\sqrt{v})] \\ C_{mac}^* &= \frac{\pi}{2} \sin[\alpha(1 + v + 2\sqrt{v})] \left[\frac{5 + 5v - 6\sqrt{v}}{16} \right] \end{aligned} \quad (11)$$

The lift and the quarter-chord moment per unit span are then given by

$$\begin{aligned} L(y) &= 0.5\rho V(y)^2 c C_l^* + \frac{\pi}{4} \rho c^2 [\ddot{\xi} + V_\infty \dot{\alpha} - (x_a - 0.25)c\ddot{\alpha}] \\ M(y) &= 0.5\rho V(y)^2 c^2 C_{mac}^* + \frac{\pi}{4} \rho c^2 \left\{ V_\infty \dot{\xi} + \frac{(x_a - 0.25)c\ddot{\xi}}{2} + V_\infty^2 \alpha \right. \\ &\quad \left. - c^2 \left[\frac{1}{32} + (x_a - 0.25)^2 \right] \ddot{\alpha} \right\} \end{aligned} \quad (12)$$

where $\theta(y)$ is the twist angle, ρ denotes the density of air, and ξ is the transverse displacement of the wing due to flapping. Furthermore, $V = \|V\|$ is the local wind speed with V defined in Eq. (8), and V_∞ is the freestream speed of the aircraft. The last term of each expression was added to Goman's original model [36] and corresponds to the apparent mass effect [38].

There is, unfortunately, no simple expression for the sectional drag coefficient. Assuming laminar flow on the wing, the sectional drag coefficient can be written as $C_D = 0.89/\sqrt{Re} + (1/\pi e A_R) C_L^2$ where A_R is the aspect ratio of the wing, $Re = \rho c V_\infty / \mu$ is the chordwise Reynolds number, and e is Oswald's efficiency factor. A refined model for calculating drag, incorporating dynamic stall, can be found in DeLaurier [38].

IV. CPG-BASED CONTROL RESULTS OF ROBObAT

Previous numerical results have shown that, for longitudinal modes, dimensional reduction via CPGs can be effective [5]. It was shown that control could be reduced to just two parameters: frequency (corresponding with velocity) and the phase difference between flapping and lead-lag ($\Delta_{31} = \Delta_{32} + \Delta_{21}$ corresponding with pitch state). Recently, open-loop nonequilibrium steady-state experiments have supported this idea further [33]. This chapter intends to show how the CPG structure allows very simple top-level controllers to provide stability and control in closed loop.

Simple symmetric proportional-integral-derivative (PID) controllers were used for all experiments,

$$\Delta_{31} = \Delta_{75} = -5(\theta - \theta_d) - 0.5\dot{\theta} - 0.1 \int_0^t (\theta - \theta_d) dt \quad (13)$$

where θ is the pitch angle and $\dot{\theta}$ is computed using the derivative filter $\omega_{cf}^2 s / [s^2 + 2\zeta_f \omega_{cf} s + \omega_{cf}^2]$, with $\omega_{cf} = 40\pi$ and $\zeta_f = 0.9$. The saturation values were set so $\Delta_{31} \in [180 \text{ deg}, 270 \text{ deg}]$. Even though we are able to use simple PID controllers in the top level, the overall controller is very nonlinear due to the CPGs in Eq. (2).

We began experimentation at an open-loop flapping frequency of 2.5 Hz. At this frequency, the open-loop appeared stable. Figure 4 shows the response to a change in desired body pitch from -10 to -20 deg. Two notes from [33] should be kept in mind. First, the actual value of body pitch is affected by the precise position of the pitch counterweight. Therefore, it is not worrisome that

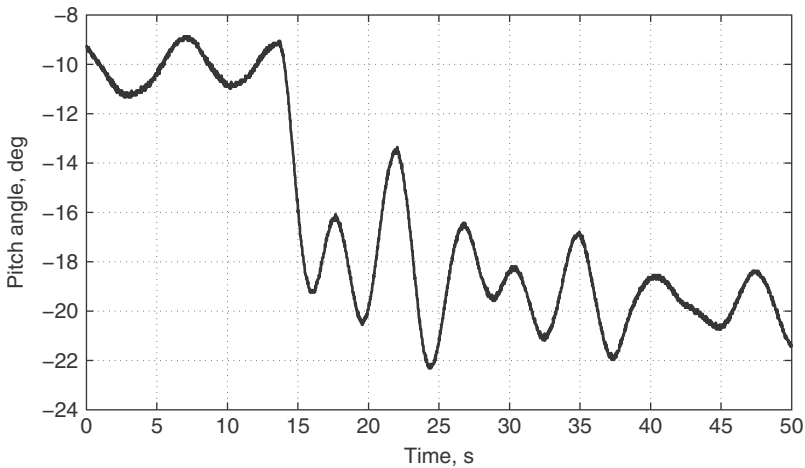


FIG. 4 Experimental results of pitch control at 2.5 Hz.

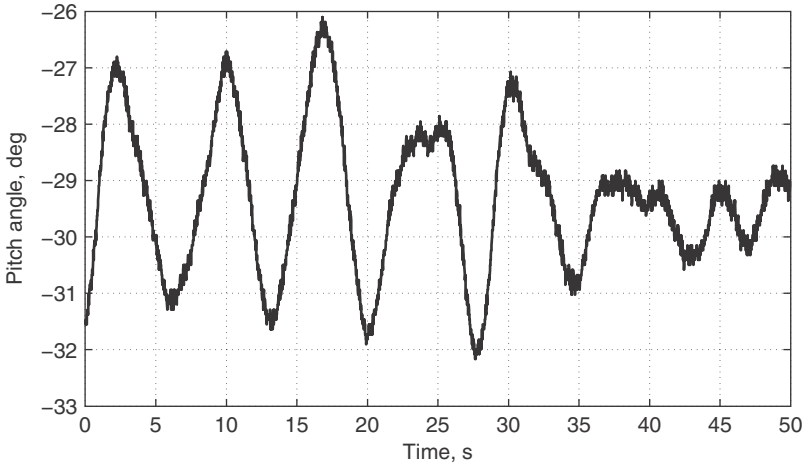


FIG. 5 Experimental results of pitch stability by control at 3 Hz.

the values are not exactly around zero or some other intuitively desired value. Second, at 2.5 Hz, the apparent maximum change of body pitch due to open-loop control of Δ_{31} is around 10–12 deg. This experiment demands a change of 10 deg and experiences saturation problems as it nears the final desired state.

Moving the frequency to 3 Hz caused instability in the open-loop. Figure 5 shows that by activating the PID control of Δ_{31} , we can stabilize the unstable system. At this frequency, we also have appreciably more control authority.

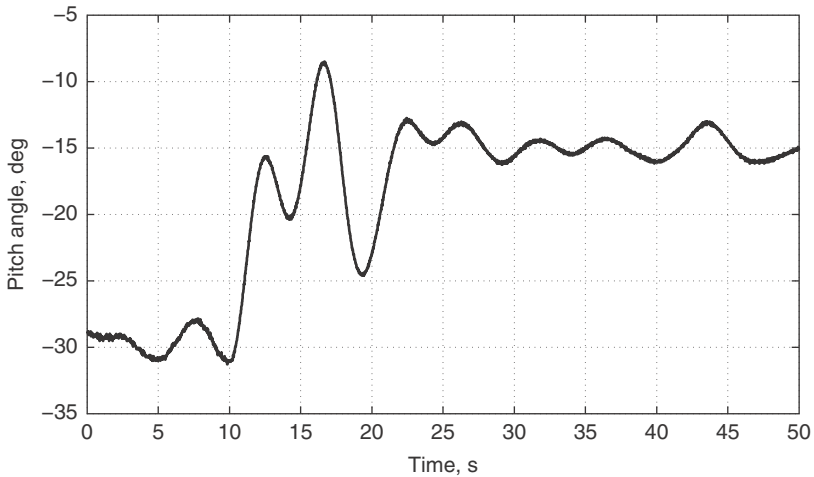


FIG. 6 Experimental results of pitch control at 3 Hz.

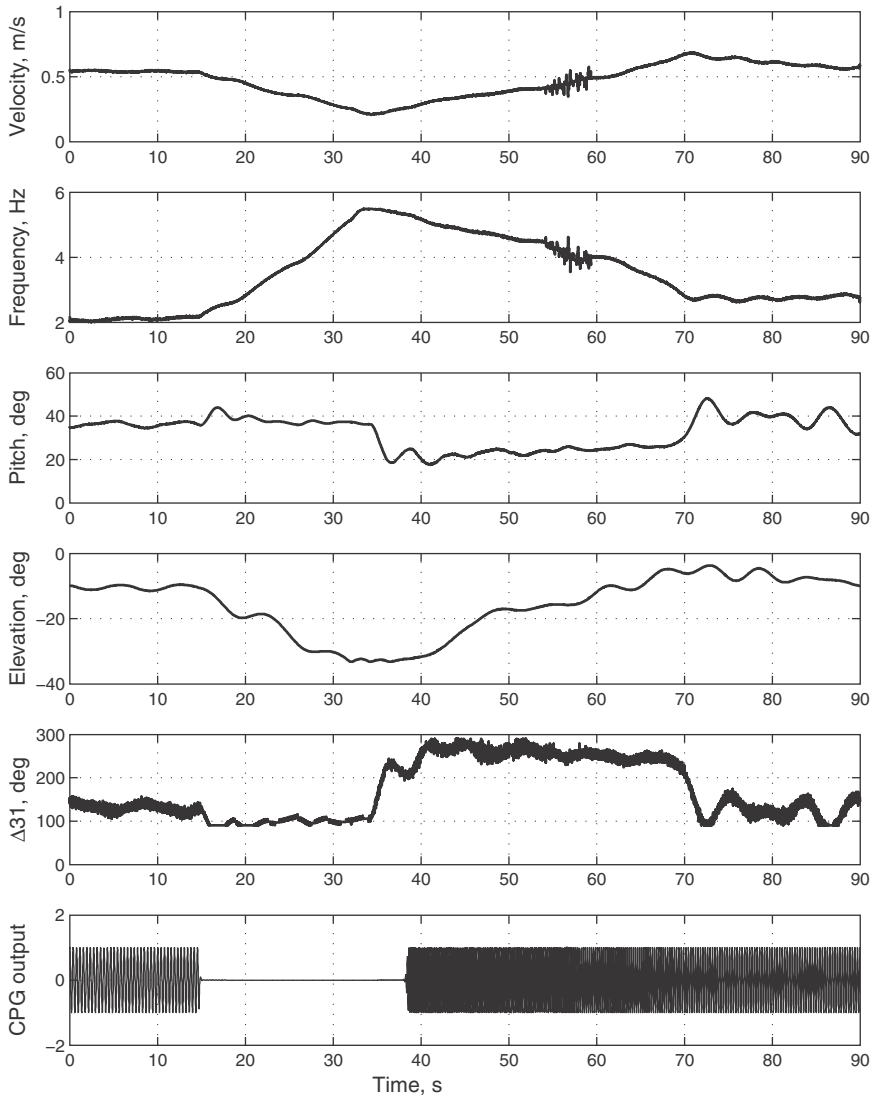


FIG. 7 Experimental results of flight mode change with pitch and velocity control.

Figure 6 shows a commanded pitch change of 15 deg, which is easily obtained. We expect that at speeds typical of bat flight (2–3 m/s with frequencies of 7–10 Hz [16]) and pitch moment of inertias not inflated by the pendulum setup we will see even more control effectiveness. Numerical results have supported the idea that this control effectiveness will be much higher [5].

Finally, we would like to feed back velocity into flapping frequency and demonstrate flight mode changes. Velocity control is achieved using a similarly simple PID controller on top of the CPG network. Importantly, we were able to transition to a glide by simply flipping the sign of the bifurcation parameter σ in Eq. (1). After dissipating energy in the glide, the transition back to flapping was again as simple as flipping the sign. Experimental results are found in Fig. 7. During the flapping phase, both pitch and velocity controllers are active (desired velocity: 0.5 m/s, desired pitch: 35 deg). During the glide phase, notice that rapid inhibition causes the CPG outputs to go to zero. (The plotted CPG output is the normalized value of the flapping angle.) At this time, the frequency and phase difference values are not meaningful, as the CPGs exhibit exponentially stable fixed-point dynamics. During such a glide, control laws like those proposed in the second part of this paper (Secs. V–VII) or in [2] could easily be used to control the bias signals of the Hopf oscillator network. A perching maneuver is not currently feasible for the flapping testbed, as the moments of inertia needed to facilitate future wing motion optimization are too large to allow a noticeable perching type maneuver. To demonstrate a perching-type maneuver, we move to a vehicle that has the appropriate weight characteristics.

V. FLIGHT MECHANICS OF MAV WITH ARTICULATED WINGS

The work presented in this paper is based on [2], where the concept of dihedral-based control for MAVs was described and analysed extensively. A few important observations have been recapitulated in this section. The word “tailless” only implies the absence of a vertical tail. Figure 8 illustrates the physics underlying the use of wing dihedral as a control. Increasing the wing dihedral reduces the force acting in the body z direction and generates a side force. The reduced z

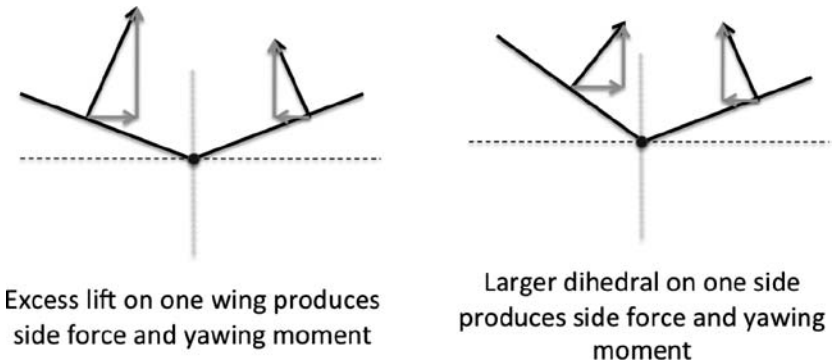


FIG. 8 Physics underlying the use of wing dihedral for control.

force affects the flight path angle of the aircraft, as well as the angle of attack and, with it, the flight speed. On the other hand, the side force can be used for providing the centripetal force for turning and as a source of the yawing moment. In particular, if the c.g. is located behind the line of action of the side force, then a positive side force produces a positive yawing moment and vice versa. It follows that a positive rolling moment (wherein the lift on the left wing is higher than the right wing) is accompanied by a positive yawing moment if the wings have a positive dihedral deflection. Consequently, the adverse yaw produced by rolling is reduced.

The yaw control effectiveness of the wing dihedral (measured in terms of the yawing moment produced per unit antisymmetric deflection) is tempered by the negative pitching moment produced by wings with a positive camber. Let δ_{asym} denote the antisymmetric deflection, that is, $\delta_{\text{asym}} = \delta_L - \delta_R$, where δ_L and δ_R denote the dihedral angles of the left and the right wing, respectively. The yaw control effectiveness $N\delta_{\text{asym}}$ is approximated as follows [2]:

$$N\delta_{\text{asym}} \approx \frac{1}{2I_z} \rho V_\infty^2 S_{\text{out}} c \left(\frac{x_a C_{L\alpha} \alpha}{c} + C_{m,\text{ac}} \right) \quad (14)$$

where x_a/c denotes the normalized distance of the aerodynamic center (AC) from the center of gravity, S_{out} is the combined area of the articulated segments (in our case, the outboard 60%) of both wings, and I_z is the aircraft moment of inertia about the z axis. The term x_a is positive if the wing aerodynamic center is ahead of the center of gravity. The effectiveness can be negative at low angles of attack for wings with positive camber ($C_{m,\text{ac}} < 0$). Thereafter, for a range of angles of attack, control effectiveness is sensitive to the angular rates before it becomes positive uniformly across the routinely flown flight envelope. The angle of attack at which the effectiveness ceases to be negative increases with increasing wing camber. The reader may be tempted to assume that the issue of negative control effectiveness only affects controllability and can be dealt with as such. However, it can have a significant impact on the turning performance of the aircraft. At low angles of attack, for example, an entry into right turns requires that the left wing dihedral be larger than the right wing dihedral to generate the required positive side force. This configuration, however, produces a negative yawing moment, which inhibits the turn. The only way to address this problem effectively is to use wing twist or ailerons. At the same time, it must be noted that controlling the wing dihedral deflections is sufficient to ensure stabilization and yaw rate regulation.

Finally, it is worth recalling that the absence of a vertical tail renders the lateral-directional dynamics unstable. The divergence for the MAV in this paper is rapid, with a time constant of approximately 0.2 s. One of the key differences between aircraft with and without a vertical tail is the buildup of roll rate. The dihedral-based mechanism described here can bring about rapid changes in the yaw rate, but it is significantly less effective at regulating the roll rate.

VI. CONTROL LAW DESIGN

A control law design for the MAV is described in this section. The control law has a two-tier hierarchical structure based on timescale separation [39], which occurs naturally between the fast rotational dynamics and the slow translational dynamics:

- The innermost loop commands the elevator and the asymmetric components of the wing dihedral.
- The outer loop commands the angle of attack and turn rate to be tracked by the inner loop based on flight speed and turn rate. The turn rate and the flight path angle are computed based on position measurements.

A schematic of the controller has been shown in Fig. 9.

A. SIMULATIONS

PID controllers can be used for controlling the MAV, with gains tuned using an approach inspired by dynamic inversion (DI) [40]. The time histories in Figs. 10 (no external disturbances) and 11 (persistent periodic disturbances) show that the controllers performed satisfactorily when the angle of attack was kept above 11 deg to ensure that the yaw control effectiveness of the dihedral was uniformly positive. A similar performance was seen for angles of attack less than 6 deg, where the yaw control effectiveness was uniformly negative.

The purpose of the simulations was to demonstrate a general control design technique. However, in the course of experiments, we were able to make reasonable estimates of the open-loop dynamics. This allowed us to tune controllers without resorting to a DI-inspired scheme.

B. ANGLE-OF-ATTACK CONTROL

The stability of the longitudinal dynamics depends on the c.g. location. Two longitudinal controllers were designed: one for the configuration with the vertical tail where the c.g. was placed around the quarter-chord point of the wing, and another for the configuration without a vertical tail where the c.g. was placed between 0.25 c and 0.3 c behind the wing AC. Here, c denotes the wing root chord length.

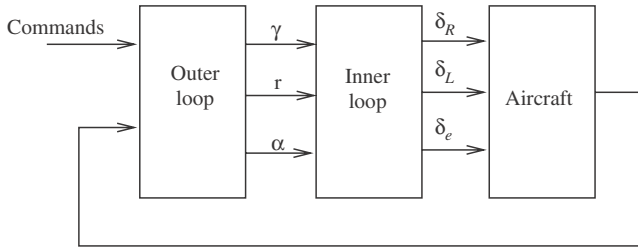


FIG. 9 Schematic of the controller, where χ denotes the aircraft heading.

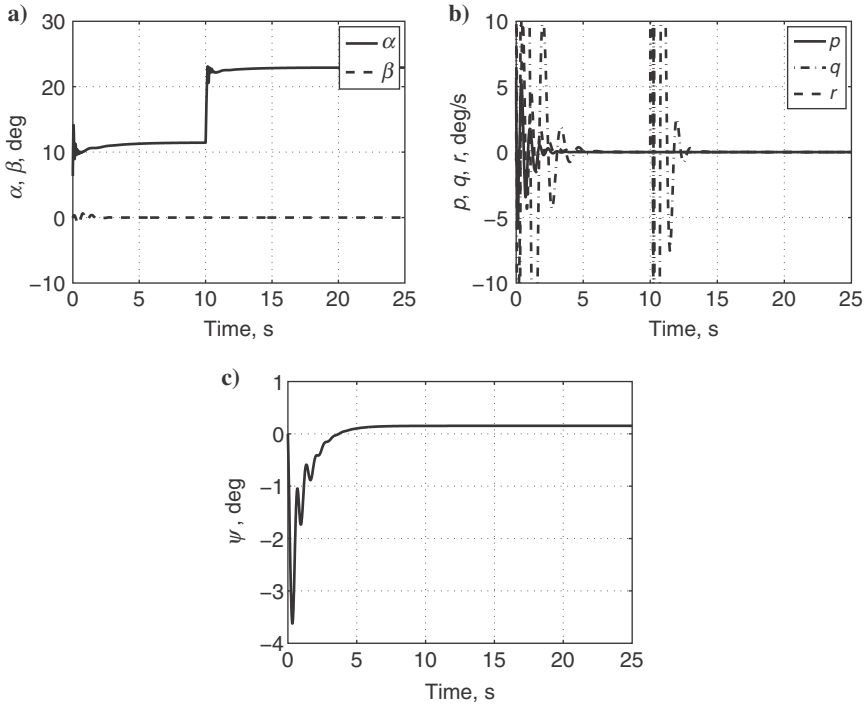


FIG. 10 Simulated time histories of the aircraft in a disturbance-free flight: a) wind axis angles α and β ; b) angular rates p , q , and r ; and c) heading angle ψ . [A 12-deg (0.2-rad) jump in the angle of attack α is commanded. The resulting disturbances are rejected by the control law.]

The longitudinal dynamics of the configuration with a vertical tail were seen during experiments to be stable across the angle-of-attack envelope, as a consequence of a favorable c.g. location, while the lateral dynamics showed a divergent unstable yaw mode. The angle of attack is controlled using a simple PID scheme, which ensures satisfactory tracking and retains an ease of implementation on the hardware.

Let $e_\alpha(t) = \alpha_c(t) - \alpha(t)$, where $\alpha_c(t)$ is the commanded angle of attack. A gain-scheduled PI controller commands the elevator deflection in the configuration with a vertical tail:

$$\delta_e(t) = k_p e_\alpha + k_i \int_0^t e_\alpha dt \quad \text{where } k_p = k_i = -0.45 - 0.0061(\alpha - 10)^2 \quad (15)$$

The longitudinal dynamics of the tailless configuration are stable, but poorly damped for $\alpha > 8$ deg. Around $\alpha = 15$ deg, the elevator effectiveness saturates,

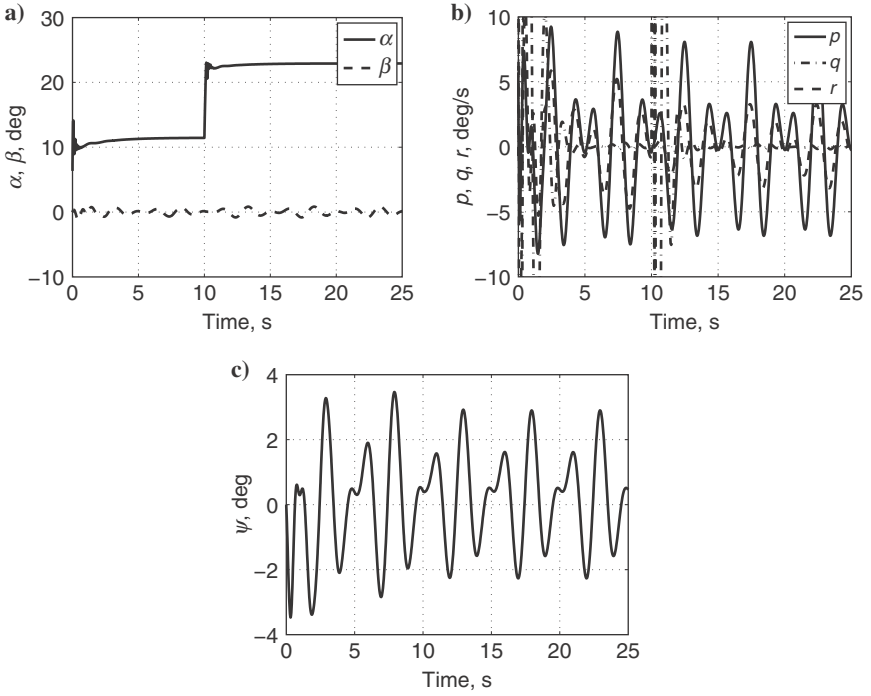


FIG. 11 Simulated time histories of the aircraft in a persistent periodic lateral-directional disturbance field: a) wind axis angles α and β ; b) angular rates p , q , and r ; and c) heading angle ψ . [A 12-deg (0.2 rad) jump in the angle of attack α is commanded.]

and higher angles of attack are unattainable under routine flight conditions. The open-loop response was measured to have a time period of 1 s. The observed reduction in the amplitude of oscillations was used to approximate the damping coefficient to 0.046. The open-loop dynamics can be written in the form

$$\ddot{\alpha} + 0.62\dot{\alpha} + 40\alpha = -40\delta_e + 5.6 \quad (16)$$

Therefore, an essentially derivative-integral controller was designed for the tailless configuration:

$$\delta_e(t) = 0.14 - \alpha_c + k_d \dot{e} + k_i \int_0^t e_\alpha dt \quad (17)$$

where the offset of 0.14 rad was added based on the measured $\delta_e - \alpha$ trims. The gain k_i was similar to that for the configuration with a vertical tail, while $k_d = 0.217$ is chosen so that the damping coefficient is approximately equal to 0.7.

C. YAW CONTROL BY ASYMMETRIC WING DIHEDRAL

Yaw control has been often neglected in the literature on perching, primarily because the aircraft possessed the traditional roll and yaw control surfaces. On the other hand, lateral-directional control is an important concern for aircraft that lack a roll control surface and use a highly unconventional yaw control mechanism. Two different yaw controllers are needed for the configurations with and without a vertical tail because the wing dihedral plays a separate role in each configuration. Moreover, although both configurations are laterally unstable, the nature of the instability is different.

In the configuration with a vertical tail, the role of the wing dihedral is to primarily provide the side force required to sustain a turn. The yaw moment required for trimming comes from the wing dihedral as well as the vertical tail. Furthermore, because the vertical tail is not actuated, the dihedral angles need to be controlled for different maneuvers such as entering or recovering from a turn.

The asymmetric component of the wing dihedral angles δ_{asym} is commanded by a PI controller. Let $e_r(t) = r_c(t) - r(t)$, where $r_c(t)$ is the commanded yaw rate. The antisymmetric dihedral deflection commanded by the controller is given by

$$\delta_{\text{asym}}(t) = 1 e_r(t) + 0.5 \int_0^t e_r(t) dt \quad (18)$$

Unlike the configuration with a vertical tail, the tailless aircraft is seen to be highly unstable in the open loop. The lateral-directional dynamics are primarily underdamped, which mandates the use of a derivative controller (unlike the PI which sufficed for the configuration with a vertical tail).

Based on experimental observations, it was estimated that the open-loop yaw-rate dynamics are of the form

$$\ddot{r} + 2\zeta\omega\dot{r} + \omega^2 r = N_{\delta_{\text{asym}}} \delta_{\text{asym}} \quad \zeta \approx -0.1, \quad \omega \approx 2\pi \quad (19)$$

for $\alpha < 8$ deg. Thereafter, the yaw dynamics are unstable and oscillatory in nature. Recall the approximation for $N_{\delta_{\text{asym}}}$:

$$N_{\delta_{\text{asym}}} \approx \frac{1}{2I_z} \rho V_\infty^2 S_{\text{out}} c \left(\frac{C_{L_\alpha} \alpha}{3} + C_{m,ac} \right)$$

where S_{out} is the combined area of the outboard sections of the two wings and I_z is the aircraft moment of inertia about the z axis. Substituting the estimates for the geometric and aerodynamic terms, it follows that

$$-2 < N_{\delta_{\text{asym}}} < -1.2 \quad \alpha < 6 \text{ deg} \quad (20)$$

Finally, in order to account for the actuator time delay of 0.2 s, a lead compensator $L(s)$ is designed given by $L(s) = 8(s + 4.5)/4.5(s + 8)$. Furthermore, a derivative filter of the form $D(s) = 12(s + 4)/s + 8$ is designed. The role of dihedral

control is regulation, and it suffices to use a derivative controller for damping addition, so that the commanded dihedral deflection is given by

$$\delta_{\text{asym}} = k_d D(s) L(s) e_r(s) \quad (21)$$

D. PERCHING GUIDANCE LOOP

The outer control loop is designed to ensure rapid changes in the flight path over a short duration. For the sake of completeness, it must be noted here that, in general, the guidance loop commands the flight path angle as well as the turn rate. The flight path angle γ , the heading angle χ , and the turn rate ω are given by [2]

$$\sin \gamma = \cos \alpha \cos \beta \sin \theta - \sin \beta \sin \phi \cos \theta - \sin \alpha \cos \beta \cos \phi \cos \theta \quad (22)$$

$$\begin{aligned} \sin \chi \cos \gamma &= \cos \alpha \cos \beta \cos \theta \sin \psi + \sin \beta (\sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi) \\ &+ \sin \alpha \cos \beta (\cos \phi \sin \theta \sin \psi \sin \phi \cos \psi) \end{aligned} \quad (23)$$

$$\omega = \dot{\chi} = \text{sign}(\dot{\chi}) \sqrt{p^2 + q^2 + r^2} \quad (24)$$

The flight path angle is controlled in discrete time so that a symmetric dihedral angle is commanded every 0.2 s (which is equal to the dihedral actuator time delay). The commanded dihedral angles are given by

$$\delta_R = \delta_L = \sqrt{2 + \frac{2}{f(\alpha) \tan \gamma_c}} \quad f(\alpha) \approx \frac{C_L(\alpha)}{C_D(\alpha)} \quad (25)$$

where γ_c is the commanded flight path angle, which is, in turn, given by

$$\gamma_c = \tan^{-1}(h) \approx \frac{h}{1 + 0.28125h^2}, \quad h = \frac{z - z_l}{\sqrt{(x - x_l)^2 + (y - y_l)^2}} \quad (26)$$

Here, x_b , y_b , and z_l are the coordinates of the desired landing point on the ground or a point in the air where a perching command is to be sent to the aircraft. The dihedral and flight path angles are computed and commanded every 0.2 s. This is not an optimal gliding strategy because it does not take into account the instantaneous flight path angle and aircraft speed. It was seen to be effective over the short duration of the experiments, although it needs to be improved for experiments that may last over a longer duration. Changing the wing dihedral brings about a significant effect in the pitching moment, and using a continuous-time flight path controller leads to undesirable oscillatory behavior due to coupling with the pitch dynamics. This problem is bypassed by updating the dihedral angle every 0.2 s, an interval that was arrived at after trial and error in the course of experiments.

VII. EXPERIMENTS ON THE ELEMENTS OF PERCHING

A perching maneuver consists essentially of two phases [20, 24]: 1) a guided approach to a suitable point away from the landing point and 2) a pitch-up, with the aircraft attaining poststall angles of attack, which ends in a perched landing at the desired point with a very low speed. The low speed allows for a perched landing, or it allows for capture mechanisms like hooks and suction pads to mechanically stop the aircraft at the desired spot [24, 41]. This section brings together the elements required to execute a perching maneuver: 1) calculations for the suitable point where the pitch-up may be commenced, 2) demonstration of control laws to hold the angle of attack and flight path during the first phase, and 3) the actual pitch-up.

A. ELEMENTS OF PERCHING

A typical perching maneuver has a two-step profile [20, 24]: a low- α glide followed by a rapid, transient pitch-up to poststall α , which achieves the desired deceleration and flattening of the flight path. A perching maneuver requires three key ingredients: 1) a guidance law that brings the aircraft to any desired point, 2) a yaw controller that regulates the heading, and 3) identification of a suitable point to commence the pitch-up. Task 1 is a formidable problem in its own right and has not been addressed here because our objective has been to understand the capabilities of the aircraft and the flight mechanics underlying the maneuvers. Task 2 was addressed in the preceding section. Task 3 has been largely ignored in the literature because perching has been studied using a stable aircraft. However, tailless aircraft are unstable, and the instability is fast enough to be of relevance even in a rapid maneuver like perching. We have addressed the problem of yaw control in Sec. VII.C.

In this section, we identify a suitable point with reference to the landing point at which the pull-up maneuver is executed. Our identification is purely at the level of flight mechanics. The reader is referred to [21, 22, 24] for optimal guidance laws. We assume that C_L is essentially constant during the second (constant δ_e) phase of perching. This leaves us with three variables to contend with: the initial flight speed, the initial flight path angle, and the distance from the landing point at which the maneuver is commenced. We seek to calculate the final speed and the C_L required for the maneuver. Note that the value of C_L required for the maneuver depends on the initial distance from the landing point.

We start with the longitudinal equations of motion of the aircraft. Let $\eta = \rho S / (2m)$, where S denotes the area of the wing, and let $\cos \gamma \approx 1$. Then, the equations of motion are given by

$$\begin{aligned}\dot{V} &= -g \sin \gamma - \eta V^2 C_D \\ \dot{\gamma} &= \eta V C_L - \frac{g}{V} \\ \dot{z} &= -V \sin \gamma\end{aligned}\tag{27}$$

We wish to use the x coordinate as the independent variable instead of t . Let $V' = dV/dx$, and note that $\dot{x} = V \cos \gamma$. We make a small angle approximation, that is, $\sin \gamma \approx \gamma$. Therefore, we get

$$\begin{aligned} V' &= -\frac{g}{V} \gamma - \eta V C_D \\ \gamma' &= \eta C_L - \frac{g}{V^2}, \quad z' = -\gamma \end{aligned} \quad (28)$$

The equation for V' can be solved analytically. Multiplying both sides by V gives

$$\begin{aligned} VV' &= -g\gamma - \eta V^2 C_D \Rightarrow \frac{V^2}{2} = e^{-2\eta C_D x} \frac{V_0^2}{2} - g \int_0^x e^{-2\eta C_D(x-\tilde{x})} \gamma d\tilde{x} \\ &\Rightarrow \frac{V^2}{2} \approx e^{-2\eta C_D x} \frac{V_0^2}{2} - g(z - z_0) \end{aligned}$$

If the pitch-up is assumed to start at $x = 0$ and $z = z_0$, then the approximate final velocity V_f at $z = z_0$ and $x = x_f$ is given by

$$V_f = e^{-\eta C_D x_f} V_0 \quad (29)$$

where V_0 is the flight speed at the time of commencement of the pitch-up.

It now remains to find an expression for C_L , which would yield C_D to compute V_f . Consider the last two equations in Eq. (28). It follows that

$$z'' = \frac{g}{V^2} - \eta C_L \quad (30)$$

We wish to command a constant value for C_L . To get an estimate, we could assume that V is a constant. This is not very accurate, especially because perching usually involves a considerable deceleration. However, because the purpose is to obtain a simple yet reliable estimate, we could use the average speed $V_c = 0.5(V_f + V_0)$. In any case, the right-hand side of Eq. (30) is a constant, and it follows that

$$z_f = 0 = \left(\frac{g}{V_c^2} - \eta C_L \right) \frac{x_f^2}{2} - \gamma_0 x_f \quad (31)$$

Furthermore, the value of C_L has to be chosen to satisfy

$$C_L = \frac{g}{\eta V_c^2} - \frac{2\gamma_0}{\eta x_f} \quad (32)$$

The right-hand side depends on $V_c = 0.5(V_f + V_0)$, where V_f is obtained from Eq. (29), which depends, in turn, on C_L . Therefore, an iterative procedure is required to compute C_L . Equation (29) also sets a bound on the smallest attainable V_f without stalling: $V_{f,\min} = e^{-\eta C_{D,s} x_f} V_0$, where $C_{D,s} = C_{D_0} + k C_{L,\max}^2$.

Figure 12 shows the C_L required as a function of γ_0 for different x_f . The value of C_L is seen to saturate for steeper flight path angles. At the same time, the final flight speed decreases with steeper initial flight path angle and a larger x_f . This observation can be explained by the fact that, in both cases, a larger distance is available for deceleration. Note, however, that once the saturation point is reached, a steeper glide renders the desired landing point unattainable. Wing twist can be used to execute a perching maneuver when the option of dropping below the landing point during the course of the pull-up is not available [42].

Figure 12 illustrates the importance of perching in the poststall flight regime, which is marked by high values of both C_L as well as C_D , which, in turn, help ensure a reduced V_f . The problem of flying in the poststall regime, though, is the possibility of loss of elevator effectiveness. This can be mitigated by using wing twist that allows the wing angle of attack to be increased to poststall values without compromising the effectiveness of the horizontal tail [43]. Wing twist can be scheduled to hold the angle of attack of the wing constant, and the elevator can be commanded to maintain the angle of attack of the aircraft (the fuselage, to be precise). The availability of wing twist, together with high drag, allows for considerably large deceleration in a very short period of time following a steep, rapid glide.

Figure 12 can also explain the importance of wing articulation. The ability to change the wing dihedral symmetrically can be used to fly steep glides without a steep increase in the flight speed. This ability would translate to a reduction in the terminal speed at the end of a perching maneuver. Note, however, that the amount of space available below the landing point constrains γ_0 , the flight path angle at the end of the descent phase. A perching maneuver that starts with a steep drop is

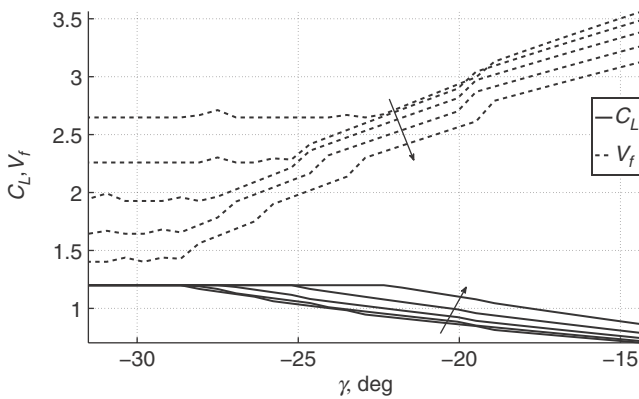


FIG. 12 This is the C_L required for perching, starting with $V_0 = 5$ m/s for different values of γ_0 and x_f . The value of C_L has been calculated using Eq. (32). The arrows indicate the direction of increasing x_f .

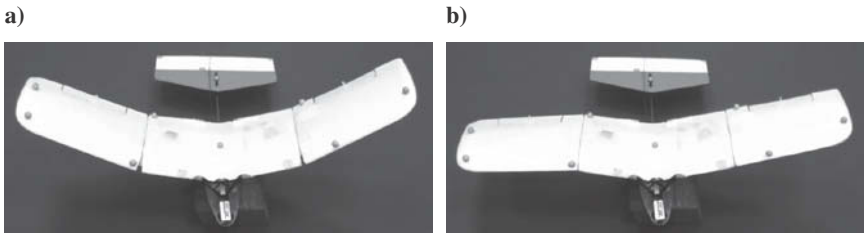


FIG. 13 Representative configurations showing the asymmetric dihedral capability of the wings: a) symmetric dihedral 40 deg and b) asymmetric configuration. (The foam table on which the aircraft is resting is not part of the airframe.)

used by birds to perch on cliffside nests and can be used by robotic aircraft to perch on power lines or ledges.

B. AIRCRAFT AND DATA ACQUISITION

The experiments described in this chapter were performed on a test MAV, shown in Fig. 13, which is a modified version of the commercially manufactured Park-Zone Ember 2. The geometric properties of the MAV are listed in Table 1. Both wings were free to rotate from a maximum 45-deg dihedral to minimum –15-deg anhedral for a total arc range of 60 deg. The actuators for wing dihedral, it may be recalled, are controlled independently on both wings for yaw stability and control.

The VICON motion-capture system, consisting of 16 infrared cameras, was used to collect flight data, in particular the aircraft position and spatial orientation,

TABLE 1 PHYSICAL PROPERTIES OF THE MAV

Property	Metric Measurement
Mass	44.0 g
Wing span	41.8 cm
Wing chord (at root)	9.5 cm
Wing incidence angle	6.0 deg
Wing dihedral	Controlled variable
MAV length	35.0 cm
Elevator area	39.12 cm ²
Propeller thrust	39 g

at 100 Hz. The VICON system uses triangulation to locate the markers with an accuracy of 1 mm. The real-time datastream provided by the VICON motion-capture system includes the global reference position and the Euler angles of each object. The availability of tracking data is contingent upon the visibility of the objects. For time steps with information loss, which were minimal and rarely comprised consecutive frames, a linear fit was used to estimate the missing data. Experiments were performed within the effective volume of capture, containing an area of 6×4 m and a height of 2 m. Because VICON provides only position and attitude information, a second-order Lagrangian polynomial was used to compute velocities and angular rates, which were then filtered to eliminate noise.

One of the limitations in the MAV is the time lag in the actuator response. For example, the actual response of the wing dihedral angle and the elevator lags the commanded values by approximately 0.2 s. Furthermore, the digital filters implemented to compensate for the time delay amplify noise in the output and are designed with a low-order Padé approximation. Because of torque limitations of the servos and their limited ability to handle high wing-loading, the dihedral angles are typically 10–15 deg higher than the commanded values. Finally, we have no way of dealing with the problem of control reversal [i.e., changing sign of $N_{\delta_{\text{asym}}}$ in Eq. (14)] in a way that accommodates the large time delay.

C. ANGLE-OF-ATTACK CONTROL FOR PERCHING

Figure 14 shows the experimentally measured longitudinal flight parameters. For these experiments, the wing dihedral was not controlled actively, which caused the aircraft heading to deviate steadily from a straight line. An angle of attack of 5 deg was commanded while the flight speed and flight path angle were not controlled.

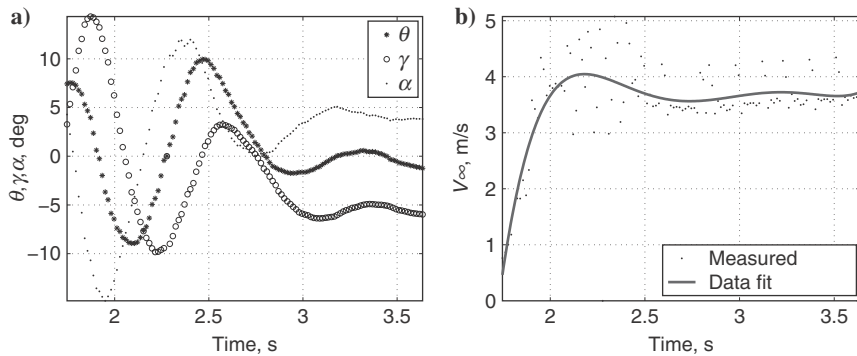


FIG. 14 Experimental results showing the longitudinal flight parameters: a) α , θ , and γ ; and b) flight speed. (In particular, α settles down at the desired value within 2 s.)

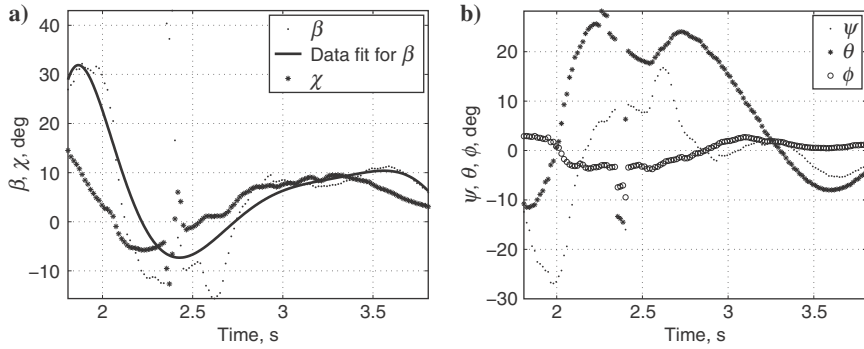


FIG. 15 Experimental results showing the sideslip, velocity heading, and the Euler angles measured during a yaw control test of the aircraft with a vertical tail. Parameters appear to be regulating during the short experiment: a) sideslip and velocity heading and b) Euler angles.

The controller for the tailless aircraft yielded similar characteristics as the one with a vertical tail.

D. LATERAL-DIRECTIONAL CONTROL FOR PERCHING

In the aircraft with a vertical tail, local lateral stability was achieved using a simple PID controller. However, in several flight tests the roll rate was seen to build up due to the dihedral effect and, without wing twist or ailerons, could not be compensated. This led to a divergent lateral-directional behavior despite local stability. Figure 15 shows the time histories for the case where the lateral dynamics were seen to be stable. A zero heading angle was commanded. The heading angle as well as sideslip converge to small values. However, the transient response does not vanish within the limited flight duration. Nevertheless, the yaw rate slows significantly by the end of the flight indicating good closed-loop stability characteristics. Lateral control of a tailless configuration is under experimental investigation.

E. FLIGHT PATH CONTROL FOR PERCHING

An effective flight path controller is necessary for a successful perching maneuver. The aircraft must be able to track the desired flight path in order to arrive at a spatial target with an acceptable flight speed and height. The PID controller gains on the angle-of-attack controller were tuned to ensure sound tracking characteristics across a range of flight path angles. The flight path angle itself, as explained in Sec. VI.D, is controlled using the wing dihedral angles. Figure 16 shows the longitudinal parameters for two experiments where flight

path angle tracking demonstrated. The flight path angle dynamics are considerably slower than the duration of the experiments, which is why a convergence is very difficult to obtain in the course of every flight test.

F. EXPERIMENTAL DEMONSTRATION OF A PERCHING MANEUVER

In conjunction with the guidance controller, a perching maneuver is executed as follows. An appropriate altitude is chosen such that a perching command is sent when the aircraft crosses it. This value was chosen to accommodate the actuation time delays for the wing dihedral as well as the elevator. Until this point, the angle-of-attack and flight-path-angle controllers described in Sec. VI were used actively. Once the aircraft reaches the prescribed altitude, zero dihedral and maximum pitch-up elevator angles are commanded. These signals are held until touchdown. Figure 17 shows the perching signal sent at the 0.6-s mark. The angle of attack builds up to 30 deg, causing the speed to reduce, and the aircraft climbs momentarily. Flight speed is halved within 1 s to 3 m/s. After a brief ascent, the MAV lands at a low angle of attack. The final speed has reduced substantially even without using wing twist. Addition of wing twist would not only enable a further reduction in the final speed, but also provide for better roll and yaw control during the approach.

Experiments are currently under way on an aircraft that uses ailerons for roll control as well as for ensuring that $N_{\delta_{\text{asym}}}$ does not change sign in the flight envelope. This will allow us to use the yaw controller during the course of the entire perching maneuver, particularly during the pull-up phase. The ailerons on either wing are controlled independently of each other, which means that they can be used as conventional flaps as well to reduce the terminal speed further.

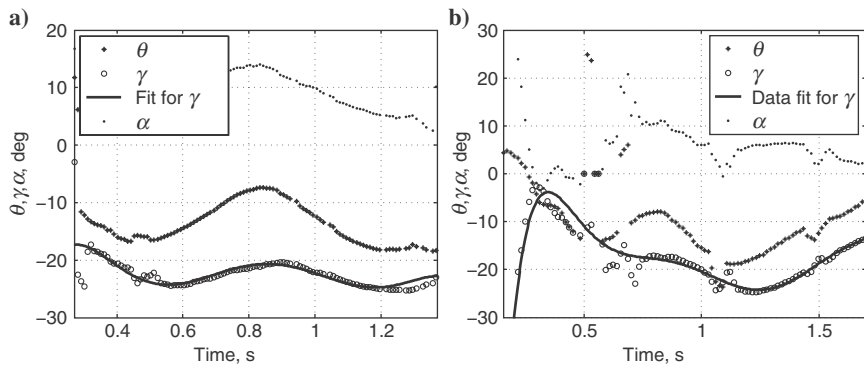


FIG. 16 Angle of attack and flight path angle during flight path guidance control: a) $\gamma_{\text{command}} = -20$ deg and b) $\gamma_{\text{command}} = -15$ deg. (The flight path angle was controlled using the wing dihedral angles.)

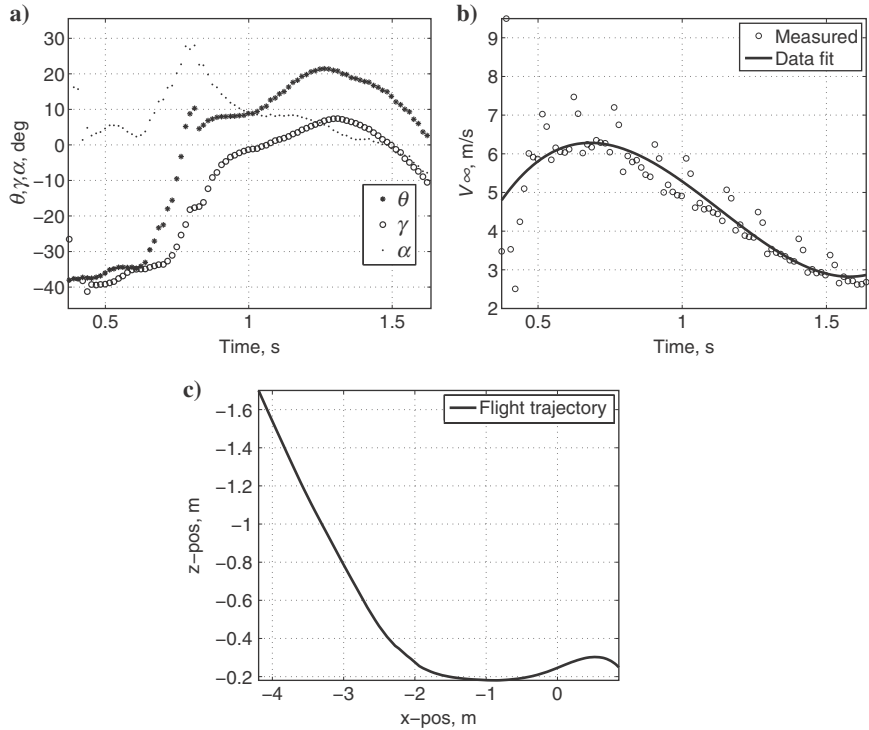


FIG. 17 Flight parameters during a perching attempt that was triggered at 1.5 m above the ground: a) α , θ , and γ ; b) flight speed; and c) trajectory.

Eventually, a claw or a suction pad may be added to the aircraft to ensure that the aircraft performs a perched landing at the desired spot.

VIII. CONCLUSION

Bat-like flight is a challenging problem that cannot be solved via averaging or with traditional tail-derived stability. We have demonstrated the ability to stabilize and control longitudinal motions via CPGs with the RoboBat. As expected, the top-level controllers are of low dimension and can be made very simple because most of the hard work is done by the CPGs. Given the extent of mechanical coupling in the design, it is remarkable that such control was immediately as effective as it was. Further work can still be done to create a pattern generator layer so as to optimize the output waveforms. Additionally, we expect to quantify the forces and moments actually produced via the aerodynamic model, so that we can make better predictions for a free-flying robotic bat.

As mechanical design of actuators develops, we expect robotic fliers in free flight to be able to utilize the key feature of phase synchronization and control of phase differences in stability and control of body motions. The major problem of identifying a method of proving such stability analytically is still open. However, this paper has demonstrated the result experimentally. Because this CPG controller design also features rapid inhibition of oscillation, it leads to the important problem of gliding flight and maneuvers while gliding.

For this reason, we described perching experiments using a novel MAV concept featuring independent wing dihedral actuation for longitudinal as well as yaw control. A guidance and control scheme was designed for the MAV, and closed-loop experiments were performed indoors to demonstrate its perching capability. Preliminary results indicate sound yaw control characteristics. Future work will focus on improving the lateral-directional control capability of the wing dihedral mechanism and adding heading tracking capability.

ACKNOWLEDGMENTS

This project was supported by the Air Force Office of Scientific Research under the Young Investigator Award Program (Grant No. FA95500910089) monitored by W. Larkin. The authors would like to thank the following students from the University of Illinois at Urbana–Champaign: Joe Kim and Nihar Gandhi.

REFERENCES

- [1] Mueller, T. J., *Fixed and Flapping Wing Aerodynamics for Micro Air Vehicle Application*, Progress in Astronautics and Aeronautics, AIAA, Reston, VA, 2001, pp. 1–10.
- [2] Paranjape, A. A., Chung, S.-J., and Selig, M. S., “Flight Mechanics of a Tailless Articulated Wing Aircraft,” *Bioinspiration and Biomimetics*, Vol. 6, No. 2, 2011.
- [3] Deng, X., Schenato, L., Wu, W. C., and Sastry, S. S., “Flapping Flight for Biomimetic Robotic Insects: Part I-System Modeling,” *IEEE Transactions on Robotics*, Vol. 22, No. 4, 2006, pp. 776–788.
- [4] Wood, R. J., “The First Takeoff of a Biologically-Inspired at-Scale Robotic Insect,” *IEEE Transactions on Robotics*, Vol. 24, No. 2, 2008, pp. 341–347.
- [5] Chung, S.-J., and Dorothy, M., “Neurobiologically Inspired Control of Engineered Flapping Flight,” *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 440–453.
- [6] Bergou, A. J., Ristroph, L., Guckenheimer, J., Cohen, I., and Wang, Z. J., “Fruit Flies Modulate Passive Wing Pitching to Generate in-Flight Turns,” *Physical Review Letters*, Vol. 104, No. 14, 2010.
- [7] Deng, X., Schenato, L., Wu, W. C., and Sastry, S. S., “Flapping Flight for Biomimetic Robotic Insects: Part II-Flight Control Design,” *IEEE Transactions on Robotics*, Vol. 33, No. 4, 2006, pp. 789–803.
- [8] Doman, D. B., Oppenheimer, M. W., and Sigthorsson, D. O., “Wingbeat Shape Modulation for Flapping-Wing Micro-Air-Vehicle Control During Hover,” *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 3, 2010, pp. 724–739.

- [9] Schenato, L., *Analysis and Control of Flapping Flight: from Biological to Robotic Insects*, Ph.D. Dissertation, Univ. of California at Berkeley, Fall 2003.
- [10] Dietl, J. M., and Garcia, E., "Stability in Ornithopter Longitudinal Flight Dynamics," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 4, 2008, pp. 1157–1162.
- [11] Leonard, B., "Flapping Wing Flight Dynamic Modeling," M.S. Thesis, Virginia Polytechnic Inst. and State Univ., Blacksburg, VA, Aug. 2011.
- [12] Grillner, S., Koslov, A., Dario, P., Stefanini, C., Menciassi, A., Lansner, A., and Kotalieski, J., "Modeling a Vertebrate Motor System: Pattern Generation, Steering and Control of Body Orientation," *Progress in Brain Research*, Vol. 165, 2007, pp. 221–234.
- [13] Kiehn, O., "Locomotor Circuits in the Mammalian Spinal Cord," *Annual Review of Neuroscience*, Vol. 29, 2006, pp. 279–306.
- [14] Dickinson, M. H., Lehmann, F. O., and Sane, S. P., "Wing Rotation and the Aerodynamic Basis of Insect Flight," *Science*, Vol. 284, June 1999, pp. 1954–1960.
- [15] Azuma, A., *The Biokinetics of Flying and Swimming*, 2nd ed., AIAA, Reston, VA, 2006, pp. 123–276.
- [16] Tian, X., Iriarte-Diaz, J., Middleton, K., Galvao, R., Israeli, E., Roemer, A., Sullivan, A., Song, A., Swartz, S., and Breuer, K., "Direct Measurements of the Kinematics and Dynamics of Bat Flight," *Bioinspiration and Biomimetics*, Vol. 1, No. 4, 2006, pp. S10–S19.
- [17] Swartz, S. M., Bishop, K. L., and Ismael-Aguirre, M.-F., "Dynamic Complexity of Wing Form in Bats: Implications for Flight Performance," *Functional and Evolutionary Ecology of Bats*, edited by Z. Akbar, G. McCracken, and T. H. Kunz, Oxford Univ. Press, Oxford, England, U.K., 2005.
- [18] Taylor, G. K., and Zbikowski, R., "Nonlinear Time-Periodic Models of the Longitudinal Flight Dynamics of Desert Locusts *Schistocerca Gregaria*," *Journal of the Royal Society Interface*, Vol. 2, No. 3, 2005, pp. 197–221.
- [19] Wang, Z. J., "Aerodynamic Efficiency of Flapping Flight: Analysis of a Two-Stroke Model," *The Journal of Experimental Biology*, Vol. 211, No. 2, 2008, pp. 234–238.
- [20] Crowther, W. J., "Perched Landing and Takeoff for Fixed Wing UAVs," *NATO Symposium on Unmanned Vehicles for Aerial, Ground, and Naval Military Operations*, RTO MP-052, 2000.
- [21] Wickenheiser, A., and Garcia, E., "Longitudinal Dynamics of a Perching Aircraft," *Journal of Aircraft*, Vol. 43, No. 5, 2006, pp. 1386–1392.
- [22] Wickenheiser, A., and Garcia, E., "Optimization of Perching Maneuvers Through Vehicle Morphing," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 4, 2008, pp. 815–823.
- [23] Roberts, J. W., Cory, R., and Tedrake, R., "On the Controllability of Fixed-Wing Perching," *Proceedings of the American Control Conference*, IEEE, Piscataway, NJ, 2009, pp. 2018–2023.
- [24] Cory, R., and Tedrake, R., "Experiments in Fixed-Wing UAV Perching," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, AIAA, Reston, VA, 2008; also AIAA Paper 2008–7256, Aug. 2008.
- [25] Paranjape, A., Guan, J., Chung, S.-J., and Krstic, M., "PDE Boundary Control for Flexible Articulated Wings on a Robotic Aircraft," *IEEE Transactions on Robotics*, 2012 (submitted for publication).

- [26] Paranjape, A., Chung, S.-J., and Krstic, M., "PDE Boundary Control for Flexible Articulated Aircraft Wings," AIAA Paper 2011-6486, Aug. 2011.
- [27] Herrero-Carrón, F., Rodríguez, F. B., and Varona, P., "Bio-Inspired Design Strategies for Central Pattern Generator Control in Modular Robotics," *Bioinspiration and Biomimetics*, Vol. 6, No. 1, 2011.
- [28] Squire, L., Berg, D., Bloom, F., du Lac, S., Ghosh, A., and Spitzer, N., *Fundamental Neuroscience*, Elsevier, New York, 2008.
- [29] Strogatz, S., *Nonlinear Dynamics and Chaos with Applications to Physics, Biology, Chemistry, and Engineering*, Perseus Books Group, Cambridge, MA, 1994.
- [30] Chung, S.-J., and Slotine, J.-J. E., "Cooperative Robot Control and Concurrent Synchronization of Lagrangian Systems," *IEEE Transactions on Robotics*, Vol. 25, No. 3, 2009, pp. 686–700.
- [31] Dorothy, M., and Chung, S., "Methodological Remarks on CPG-Based Control of Flapping Flight," AIAA Paper 2010–7634, Aug. 2010.
- [32] Chung, S.-J., Dorothy, M., and Stoner, J. R., "Neurobiologically Inspired Control of Engineered Flapping Flight," AIAA Paper 2009–1929, April 2009.
- [33] Kuang, P. D., Dorothy, M., and Chung, S.-J., "RoboBat: Dynamics and Control of a Robotic Bat Flapping Flying Testbed," AIAA Paper 2011–1435, March 2011.
- [34] Paranjape, A. A., Chung, S.-J., Hilton, H. H., and Chakravarthy, A., "Dynamics and Performance of a Tailless MAV with Flexible Articulated Wings," *AIAA Journal*, Vol. 50, No. 5, 2012, pp. 1177–1188.
- [35] Slotine, J.-J. E., and Li, W., *Applied Nonlinear Control*, Prentice–Hall, Upper Saddle River, NJ, 1991.
- [36] Goman, M., and Khrabrov, A., "State-Space Representation of Aerodynamic Characteristics of an Aircraft at High Angles of Attack," *Journal of Aircraft*, Vol. 31, No. 5, 1994, pp. 1109–1115.
- [37] Peters, D., Karunamoorthy, S., and Cao, W., "Finite State Induced Flow Models Part I: Two-Dimensional Thin Airfoil," *Journal of Aircraft*, Vol. 32, No. 2, 1995, pp. 313–322.
- [38] DeLaurier, J., "An Aerodynamic Model for Flapping-Wing Flight," *Aeronautical Journal*, Vol. 97, No. 964, 1993, pp. 125–130.
- [39] Wang, Q., and Stengel, R., "Robust Nonlinear Flight Control of a High-Performance Aircraft," *IEEE Transactions on Control Systems Technology*, Vol. 13, No. 1, 2005, pp. 15–26.
- [40] Hovakimyan, N., Lavretsky, E., and Sasane, A., "Dynamic Inversion for Nonaffine-in-Control Systems via Time-Scale Separation. Part I," *Journal of Dynamical and Control Systems*, Vol. 13, No. 4, 2007, pp. 451–465.
- [41] Anderson, M. L., Perry, C. J., Hua, B. M., Olsen, D. S., Parcus, J. R., Pederson, K. M., and Jensen, D. D., "The Sticky-Pad Plane and Other Innovative Concepts for Perching UAVs," AIAA Paper 2009–40, Jan. 2009.
- [42] Chakravarthy, A., Paranjape, A. A., and Chung, S.-J., "Control Law Design for Perching an Agile MAV with Articulated Wings," AIAA Paper 2010-7934, Aug. 2010.
- [43] Paranjape, A. A., and Chung, S.-J., "Flight Mechanics of a Tailless Articulated Wing Aircraft," AIAA Paper 2010–7633, Aug. 2010.

Neural Network-Based Optimal Control of an Unmanned Helicopter

David Nodland* and H. Zargarzadeh*

Missouri University of Science and Technology, Rolla, Missouri 65409

Arpita Ghosh[†]

National Metallurgical Laboratory, Jamshedpur 831007, India

S. Jagannathan[‡]

Missouri University of Science and Technology, Rolla, Missouri 65409

I. INTRODUCTION

Unmanned helicopters are self-propelled and autonomously controlled rotorcraft that are capable of independent flight. Because of their versatility and maneuverability, these unmanned helicopters are indispensable for both civilian and military applications, where human intervention is restricted. For control of a helicopter [1], it is necessary to produce moments and forces on the aircraft with two goals: first, to position the helicopter in equilibrium such that the desired trim state is achieved; and second, to control the helicopter's velocity, position, and orientation such that it tracks a desired trajectory with minimal error. The dynamics of the unmanned helicopter are not only nonlinear but are also coupled with each other and underactuated, which makes control difficult. In other words, the helicopter has six degrees of freedom (DOF), which must be controlled with only four control inputs—thrust and three rotational torques.

To solve the problem of controlling a rotary-wing unmanned aerial vehicle (UAV), several techniques have been proposed [1–9]. The full model of the helicopter is quite complex when the main rotor and tail rotor and effects of drag, actuator dynamics, and other disturbances are taken into consideration. Based on Newton–Euler equations, a dynamic model has been derived [1] considering the helicopter as a rigid body with input forces and torques applied to the center of mass. Also, it has been shown [1] that the multivariable nonlinear helicopter model cannot be converted into a controllable linear system via exact state-space linearization. In addition, for certain output functions, exact input-output linearization results in

*Graduate Research Assistant, Department of Electrical and Computer Engineering.

[†]Research Scientist.

[‡]Rutledge-Emerson Distinguished Professor, Department of Electrical and Computer Engineering, sarangap@mst.edu.

unstable zero dynamics [10]. However, if only the position and heading are chosen as outputs, by neglecting the coupling between moments and forces, the approximated system with dynamic decoupling is full-state linearizable, and as a result output tracking can be employed [1]. A parameterized model of the Yamaha R-50 helicopter has been identified in [2] using frequency-domain methods. The accuracy of the identified model is verified by comparing the model-predicted responses with the responses collected during flight experiments for both hover and cruise flight conditions.

In [3], adaptive output feedback control of uncertain nonlinear systems with unknown dynamics and dimensions is considered, with emphasis on its application to high-bandwidth unmanned helicopters. Under the assumption that the system is feedback linearizable, a single hidden-layer neural network (NN) has been introduced to cancel the inversion error. The NN learns online, with no off-line training required. A simple linear observer has been introduced to estimate the derivatives of the tracking error. The observer provides estimates only for the states that are feedback linearized, and not for the states that are actually associated with the internal dynamics.

For autonomous helicopter flight, the control problem [4] has been separated into an inner-loop attitude control and an outer-loop trajectory control problem. Adaptive techniques have been applied to minimize the effects of model errors on all six degrees of freedom, leading to more accurate position tracking. The pseudocontrol hedging method is used to enable adaptation to occur in the outer loop without interacting with the attitude dynamics. The underactuated helicopter system is proven to be feedback linearizable with asymptotically stable zero dynamics, and an output feedback control law has been designed for both stabilization and trajectory tracking of the helicopter dynamic model [5]. A drawback of these controllers [2–5] is that they do not provide a nonlinear online optimal approach. Also, the dynamics due to blade flapping and aerodynamic drag are simplified, and the coupling between rolling (pitching) moments and lateral (longitudinal) accelerations are neglected. It was shown in [1] that the preceding simplifications are valid only at very low speeds such as hovering, whereas the aerodynamic effects can become significant even at moderate velocities, causing instability of the helicopter.

A backstepping-based controller has been presented in [6] for autonomously landing a helicopter, which also holds for full flight control. The helicopter model includes both flapping and servo dynamics. The nonlinear controller computes the desired thrust and blade flapping angles to get the commanded position and then computes the control inputs necessary to achieve the desired thrust and flapping angles.

Tracking control [7] of a model helicopter FLYRT has been done by applying direct neural dynamic programming, which is a structured cascade of neural networks comprising the action, critic, and trim networks. The action network provides controls required to drive the helicopter, the critic network approximates the cost function if an explicit cost function does not exist, and the trim network provides

nominal trim positions as a function of desired system operating conditions. A robust command augmentation system has been designed [8] for a helicopter model with insufficient information about helicopter dynamics using an adaptive NN. It has been shown that when dynamic model inversion is applied to the helicopter model with limited knowledge of hovering dynamics, the command-following performance is degraded severely by the inversion error, in particular, during rapid maneuvers. When the adaptive neural network is applied to hovering dynamics, however, the effect of the inversion error is apparently reduced through online adaptation.

NN-based controllers trained off-line are often robust to small disturbances or modeling errors but fail to adapt to more substantial disturbances or more significant modeling errors. Further, an off-line scheme alone does not allow the NN to learn any new dynamics it encounters during a new maneuver. On the other hand, NN controllers that learn online quickly adapt to variations in the nonlinear behavior of the system in real time with no prior knowledge of the system dynamics. A nonlinear controller for a quadrotor unmanned aerial vehicle has been proposed in [9] by employing output feedback and NNs. Consideration has been given to uncertain nonlinear terms such as aerodynamic drag and blade flapping. Complete control systems have been designed and implemented [11, 12], but no proof of optimality is available with these approaches. Full control system design and flight testing were also performed by [13], but the controller was nonlinear and nonoptimal. Optimal control has been performed for a helicopter UAV, but the optimal controller is linear, rather than nonlinear [14].

In [15], optimal regulation and tracking of nonlinear discrete-time affine systems have been proposed by solving the discrete-time Hamilton–Jacobi–Bellman (HJB) equation online and forward in time. Using an initial stabilizing control, an online approximator (OLA) was tuned to learn the HJB equation while a second OLA was utilized to minimize the cost (HJB) function based on the information provided by the first OLA. Lyapunov methods were used to demonstrate that the approximated control inputs approached the optimal control inputs with a small bounded error. In addition, a single online approximator (SOLA)-based scheme has been introduced in [16] to solve the optimal regulation and tracking control problems for affine nonlinear continuous-time systems with known dynamics. The SOLA-based adaptive approach has been designed to learn the infinite horizon continuous-time HJB equation, and the corresponding optimal control input that minimizes the HJB equation has been calculated forward in time.

However, an optimal controller design for regulation and tracking of an underactuated helicopter using NNs has not yet been attempted. Following the approach given in [16], the SOLA-based scheme for optimal regulation of a nonlinear continuous-time strict feedback system with known dynamics has been considered in this chapter. The online-approximator-based dynamic controller learns the continuous-time HJB equation and then calculates the corresponding optimal control input that minimizes the HJB equation forward in time. This SOLA-based optimal control scheme is then extended for optimal tracking of an unmanned helicopter with known dynamics. The proposed tracking controller

consists of a single NN for approximating the cost function, with the NN tuned online using a weight update law. A virtual controller is also employed to provide a feedforward term needed by the optimal controller. Further, Lyapunov analysis is utilized to demonstrate the stability of the closed-loop system.

The main contribution of this chapter is the development of an optimal control-based control scheme for regulation and trajectory tracking of a helicopter UAV, forward in time, with the helicopter system expressed in a form appropriate for backstepping control. The controller tuning is independent of the trajectory, in contrast with [9]. A NN-based online approximator is utilized to approximate the cost function, and the overall stability is guaranteed. The optimal controller has been previously developed for backstepping systems, but has not yet been applied to helicopter UAVs. This optimal controller previously required that $f(0) = 0$, but the virtual controller in this work eliminates the need for that requirement. This chapter builds on [16] and [17] from the fields of optimal and helicopter control; however, this work both uses previous developments for a new application and adds to these works with a closed-loop stability proof demonstrating the proposed controls scheme's convergence with state feedback. The closed-loop proof is not direct but involved and is included near the end of the chapter.

The chapter begins by presenting the nonlinear model of the helicopter in the next section. Later sections address the kinematic controller, the virtual controller, the continuous-time nonlinear optimal HJB regulation and tracking problem, and the solution of the HJB equation forward in time, making use of a single online approximator-based optimal controller. The final sections include stability analysis for the closed-loop system, simulation results, and concluding remarks.

II. HELICOPTER DYNAMIC MODEL

Consider the helicopter shown in Fig. 1 with six DOF defined in the inertial coordinate system \mathcal{Q}^a , where its position coordinates are given by $\rho = [x, y, z]$ and its orientation described as yaw, pitch, and roll, respectively, is given by the Euler angles $\Theta = [\phi, \theta, \psi]$. The equations of motion are expressed in the body-fixed coordinate system \mathcal{Q}^b , which is associated with the helicopter's center of mass. The $^b x$ axis is defined parallel to the helicopter's direction of travel, and the $^b y$ axis is defined perpendicular to the helicopter's direction of travel, whereas the $^b z$ axis is defined as projecting orthogonally downwards from the xy plane of the helicopter. The following variables are used for the dynamics:

$m \in \mathbb{R}$ is a positive scalar denoting the mass of the helicopter,

$F \in \mathbb{R}^{3 \times 1}$ is the body force applied to the helicopter's center of mass,

$v = [v_x, v_y, v_z] \in \mathbb{R}^{3 \times 1}$ represents the translational velocity vector,

$\omega = [\omega_x, \omega_y, \omega_z] \in \mathbb{R}^{3 \times 1}$ represents the angular velocity vector,

$I \in \mathbb{R}^{3 \times 3}$ is the identity matrix, and

$\mathcal{J} \in \mathbb{R}^{3 \times 3}$ is the positive-definite inertia matrix.

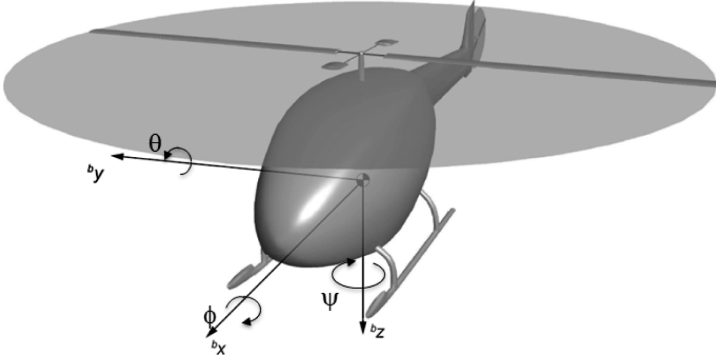


FIG. 1 Helicopter dynamics.

The kinematics of the helicopter are given by

$$\dot{\rho} = Rv \quad (1)$$

and

$$\dot{\Theta} = T^{-1}\omega \quad (2)$$

The rotational transformation matrix from body-fixed frame to the inertial coordinate frame is defined as

$$R(\Theta) = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\theta s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}$$

where s_\bullet and c_\bullet denote the $\sin(\bullet)$ and $\cos(\bullet)$ functions, respectively. Note that $\|R\|_F < R_{\max}$ for a known constant R_{\max} and $R^{-1} = R^T$. Throughout this chapter, $\|\bullet\|$ denotes a Euclidean norm, and $\|\bullet\|_F$ denotes a Frobenius norm. The translational rotation matrix is used to transition from the angular velocity to the derivative of the orientation and is defined as

$$T(\Theta) = \frac{1}{c_\theta} \begin{bmatrix} 0 & s_\psi & c_\psi \\ 0 & c_\theta c_\psi & -c_\theta s_\psi \\ c_\theta & s_\theta s_\psi & s_\theta c_\psi \end{bmatrix}$$

where t_\bullet has been used to represent $\tan(\bullet)$. The transformation matrix is bounded according to $\|T\|_F < T_{\max}$ for a known constant T_{\max} , provided $-\pi/2 < \phi < \pi/2$ and $-\pi/2 < \theta < \pi/2$ such that the helicopter trajectory does not pass through any singularities [1]. Redefining Euler angles or using quaternions would provide an alternative method for avoiding singularities, but some trajectories involving these singularities would still not be possible because of the

physical limitations of helicopters. Let the mass-inertia matrix M be defined as

$$M = \begin{bmatrix} mI & 0 \\ 0 & \mathcal{J} \end{bmatrix}$$

Now the dynamics can be written in the form given in [9], but with dynamics as given in [17]

$$M \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \bar{S}(\omega) + \begin{bmatrix} 0^{3 \times 1} \\ \mathcal{N}(\omega) \end{bmatrix} + \begin{bmatrix} G(R) \\ 0^{3 \times 1} \end{bmatrix} + U + \tau_d \quad (3)$$

where $\bar{S}(\omega) = [0^{3 \times 1}, -\omega \times J\omega]^T$, $\mathcal{N}(\omega) \in \mathbb{R}^{3 \times 1}$ represents the nonlinear aerodynamic effects such as fuselage drag, $G(R) \in \mathbb{R}^{3 \times 1}$ represents the gravity vector and is defined as $G(R) = m\bar{g}R^T$ with \bar{g} the gravitational acceleration, and $\tau_d = [\tau_{d1}^T, \tau_{d2}^T]^T$ represents unknown bounded disturbances such that $\|\tau_d\| < \tau_M$ for all time t , with τ_M a known positive constant. Note that (\times) denotes the vector cross product. The nonlinear aerodynamic effects taken into consideration for modeling of the helicopter are given by $\mathcal{N}(\omega) = Q_M e_3 - Q_T e_2$, with Q_M and Q_T aerodynamic constants for which values are given in the simulation section, and originally found in [17]. Note that e_1 , e_2 , and e_3 are unit vectors directed along the x , y , and z axes, respectively, in the inertial reference frame. U is as given here:

$$U = \begin{bmatrix} E_3^a & 0^{3 \times 3} \\ 0^{3 \times 1} & \text{diag}([p_{11} \ p_{22} \ p_{33}]) \end{bmatrix} \begin{bmatrix} u \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

where p_{ii} are positive-definite constants, with u providing the thrust in the z direction; w_1 , w_2 , and w_3 providing the rotational torques; and with $E_3^a = [0 \ 0 \ 1]^T$. Defining the new augmented variables $X = [\rho^T \ \Theta^T]^T \in \mathbb{R}^{6 \times 1}$ and $V = [v^T \ \omega^T]^T \in \mathbb{R}^{6 \times 1}$, the full dynamics can be written employing the backstepping technique in the strict-feedback form as

$$\dot{X} = AV + \xi \quad (4)$$

$$\dot{V} = f(V) + M^{-1}U \quad (5)$$

where $f(V) = M^{-1}\{\bar{S}(\omega) + [0^{3 \times 1} \mathcal{N}(\omega)]^T\} + \bar{G}$ with $\bar{G} = M^{-1}G(R) \in \mathbb{R}^{6 \times 1}$, with $\xi \in \mathbb{R}^{6 \times 1}$ the bounded sensor measurement noise such that $\|\xi\| \leq \xi_M$ for a known constant ξ_M and

$$A = \begin{bmatrix} R & 0^{3 \times 3} \\ 0^{3 \times 3} & T^{-1} \end{bmatrix}$$

Writing $f(V)$ explicitly yields

$$f(V) = \begin{bmatrix} \frac{1}{m} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\mathcal{J}_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\mathcal{J}_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{\mathcal{J}_z} \end{bmatrix} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ -\omega_x \times \mathcal{J}_x \omega_x \\ -\omega_y \times \mathcal{J}_y \omega_y \\ -\omega_z \times \mathcal{J}_z \omega_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -Q_T \\ Q_M \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m\bar{g} \\ 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Writing Eqs. (4) and (5) explicitly yields

$$\dot{X} = AV + \xi \quad (6)$$

$$V = f(V) + \begin{bmatrix} mI & 0^{3 \times 3} \\ 0^{3 \times 3} & \mathcal{J} \end{bmatrix}^{-1} \begin{bmatrix} E_3^a & 0^{3 \times 3} \\ 0^{3 \times 1} & \text{diag}([p_{11} \ p_{22} \ p_{33}]) \end{bmatrix} u_v \quad (7)$$

Although backstepping is an approach that may bring certain challenges with it, it is necessary because of the strict-feedback nature of the system dynamics. The next section will examine nonlinear optimal regulation and tracking of the helicopter.

III. NONLINEAR OPTIMAL REGULATION AND TRACKING OF THE HELICOPTER

In this section, the optimal control framework for selection of u_v is provided. Because of the NN approximating the cost function, an approximate optimal-based input \hat{u}_v is generated as seen in Fig. 2. The first part of this section introduces the kinematic controller, which generates the desired velocity for the dynamic controller. After the kinematic controller, the virtual controller is introduced in Sec. III.B to provide a feedforward term u_d for the dynamic controller. Section III.C addresses the HJB equation, providing a foundation for a NN-based optimal controller, which is discussed in Sec. III.D. These results are extended to the tracking case, which provides the optimal control term \hat{u}_e^* used for the combined NN-based input to the helicopter dynamics \hat{u}_v .

A. KINEMATIC CONTROLLER

The kinematic controller generates the desired velocity for the dynamic controller. To design the kinematic controller for the unmanned helicopter, the tracking error for the position must first be defined. The position tracking error is given by

$$\delta_1 = \rho_d - \rho \quad (8)$$

The desired velocity v_d is then determined as in [17] as

$$v_d = \hat{v} - \frac{1}{m} \delta_1$$

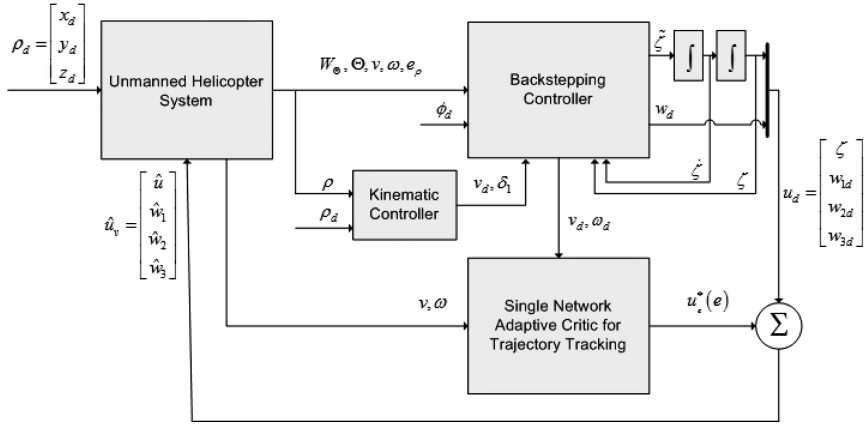


FIG. 2 Control scheme for optimal tracking.

There exist desired trajectories that may reach unstable operating regions as the orientation about the x and y axes approaches $\pm \pi/2$. This is a consequence of the physical limitations of helicopters. Therefore, trajectories requiring that these orientations be maintained should not be assigned to the helicopter.

B. VIRTUAL CONTROLLER

The virtual controller is used to obtain the virtual control output or desired input $u_d = [\zeta \ w_{1d} \ w_{2d} \ w_{3d}]^T$. The steps given here follow the approach taken in [17]. This process is performed by first defining a set of error terms. The first, $\delta_1 = \rho_d - \rho$, was introduced with the kinematic controller. The second error term to be minimized is $\delta_2 = m(v - v_d)$, with δ_2 a velocity tracking error that incorporates the helicopter's mass. The third and fourth errors to be considered are $\epsilon_3 = \phi - \phi_d$ and $\epsilon_4 = \dot{\phi} - \dot{\phi}_d$, which consider the error in the helicopter's heading and the rate at which this error is changing. A fifth error term considers the error in the thrust and can be expressed as

$$\delta_3 = m\bar{g}e_3 - m\dot{v}_d + \delta_2 + \frac{1}{m}\delta_1 - \zeta R(\Theta)e_3 \quad (9)$$

with all of the variables in Eq. (9) as defined earlier. For convenience, a term

$$Y_d = \delta_2 + \delta_3 + \frac{d}{dt} \left(m\bar{g}e_3 - m\dot{v}_d + \delta_2 + \frac{1}{m}\delta_1 \right)$$

is introduced prior to the final error term necessary for this development, which allows this final error term to be written as

$$\delta_4 = Y_d - [\dot{\zeta} R(\Theta)e_3 + \zeta R(\Theta)\text{skew}(\omega)e_3]$$

The choice of these particular error terms is analyzed in further detail in [17].

Selecting

$$\tilde{\zeta}R(\Theta)e_3 - \zeta R(\Theta)\text{skew}(e_3)\tilde{w}_d = \dot{Y}_d - 2\dot{\zeta}R(\Theta)\text{skew}(\omega)e_3 + \delta_3 + \delta_4 \quad (10)$$

to be solved for control of the main rotor thrust, pitch, and roll, and

$$\ddot{\phi} = \ddot{\phi} - \epsilon_3 - \epsilon_4 \quad (11)$$

To be solved for control of the yaw [17], a solution for both equations is given by

$$\begin{bmatrix} \tilde{w}_{1d} \\ \tilde{w}_{2d} \\ \tilde{\zeta} \end{bmatrix} = \begin{bmatrix} 0 & \zeta & 0 \\ -\zeta & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} R(\Theta)^T [\dot{Y}_d - 2\dot{\zeta}R(\Theta)\text{skew}(\omega)e_3 + \delta_3 + \delta_4]$$

from which \tilde{w}_{1d} , \tilde{w}_{2d} , and $\tilde{\zeta}$ are obtained (with $\tilde{\zeta}$ obtained recursively). This solution was obtained by making use of the property $\text{skew}(e_3)\tilde{w}_d = e_3 \times \tilde{w}_d = -\text{skew}(\tilde{w}_d)e_3$ and rearranging and rewriting Eq. (10). Defining the relationship between the angular velocity and orientation (from Sec. II) as

$$\dot{\Theta} = \frac{1}{c_\theta} \begin{bmatrix} 0 & s_\psi & c_\psi \\ 0 & c_\theta c_\psi & -c_\theta s_\psi \\ c_\theta & s_\theta s_\psi & s_\theta c_\psi \end{bmatrix} \omega = T^{-1} \omega \quad (12)$$

it is now possible to rearrange Eq. (5) in terms of $\dot{\omega}$ and set $\dot{\omega} = \tilde{w}_d$ while considering only the virtual control inputs. Doing this yields

$$\tilde{w}_d = -\mathcal{J}^{-1} \omega \times \mathcal{J} \omega + |Q_M| \mathcal{J}^{-1} e_3 - |Q_T| \mathcal{J}^{-1} e_2 + \mathcal{J}^{-1} P \tilde{w}_d$$

Taking the derivative of Eq. (12), rearranging, and considering only the yaw (first element in orientation vector) result in

$$\ddot{\phi} = -e_1^T T^{-1} \dot{T} T^{-1} \omega + \frac{1}{c_\theta} (s_\psi \tilde{w}_{2d} + c_\psi \tilde{w}_{3d}) \quad (13)$$

Then, employing both Eqs. (11) and (13) and rearranging allow \tilde{w}_{3d} to be obtained as

$$\tilde{w}_{3d} = \frac{c_\theta}{c_\psi} (\ddot{\phi} - \epsilon_4 - \epsilon_3 + e_1^T T^{-1} \dot{T} T^{-1} \omega - \frac{s_\psi}{c_\theta} \tilde{w}_{2d})$$

Now the real inputs are obtained. To do this, first restate a portion of the dynamics to obtain w_d from Eq. (5) as $w_d = P^{-1}(\mathcal{J} \tilde{w}_d + \omega \times \mathcal{J} \omega - Q_M e_3 + Q_T e_2)$ with $P = \text{diag}([p_{11} \ p_{22} \ p_{33}]^T)$ a set of gains, and then obtain ζ by double-integrating from

$$\ddot{\zeta} = \tilde{\zeta}$$

by using the value that has just been obtained for $\tilde{\zeta}$. Combining the preceding results allows one to arrive at the feedforward portion of the control input as

$$u_d = [\zeta \ w_{1d} \ w_{2d} \ w_{3d}]^T \quad (14)$$

from the values that have just been obtained for ζ , w_{1d} , w_{2d} , and w_{3d} . Proof that the inputs generated by these equations will ensure convergence is provided in [17].

C. HAMILTON–JACOBI–BELLMAN EQUATION

In this section, the optimal control input is designed based on the dynamics of the helicopter with the intention of minimizing the control input errors. The dynamics of the helicopter given is of the form

$$\dot{V} = f(V) + gu_V \quad (15)$$

where $V \in \mathbb{R}^{6 \times 1}$, $f(V) \in \mathbb{R}^{6 \times 1}$, $g \in \mathbb{R}^{6 \times 6}$ is bounded satisfying $g_{\min} \leq \|g\|_F \leq g_{\max}$, and $u_V \in \mathbb{R}^{4 \times 1}$ is the control input. For reference, g is provided here explicitly as

$$M^{-1} \begin{bmatrix} E_3^a & 0^{3 \times 3} \\ 0^{3 \times 1} & \text{diag}([p_{11} \ p_{22} \ p_{33}]) \end{bmatrix}$$

For this system, $V = 0$ is a unique equilibrium point on compact set $Y \subset \mathbb{R}^{6 \times 1}$ with $f(0) = 0$. By adding the feedforward term as will be presented later in this chapter, however, it is possible to neglect the $f(0) = 0$ requirement. Under these conditions, the optimal control input for the unmanned helicopter system given in Eq. (15) can be determined [18]. The dynamics $f(V)$ and g are assumed to be known throughout this chapter; however, this assumption can be relaxed if some of the unknown parameters are estimated by using NNs.

The infinite horizon HJB cost function for Eq. (15) is given here:

$$W[V(t)] = \int_t^\infty r[V(\tau), u_V(\tau)] d\tau \quad (16)$$

where $r[V(t), u_V(t)] = Q(V) + u_V^T B u_V$, $Q(V) > 0$ is the positive-definite penalty on the states and with $B \in \mathbb{R}^{6 \times 6}$ a positive-definite matrix. The control input must be selected such that the cost function in Eq. (16) is finite, and it is assumed that there is an admissible controller [15].

The Hamiltonian for the cost function in Eq. (16) with control input u_V is defined as

$$H(V, u_V) = r(V, u_V) + W_V^T(V)[f(V) + gu_V] \quad (17)$$

where $W_V(V)$ is the gradient of $W(V)$ with respect to V . The optimal control input u_V^* , which minimizes the cost function in Eq. (16), will also minimize the Hamiltonian in Eq. (17). Consequently, the optimal control input can be obtained by solving $\partial H(V, u_V)/\partial u_V = 0$ and is found to be as shown here:

$$u_V^*(V) = \frac{-B^{-1}g^T W_V^*(V)}{2} \quad (18)$$

Substituting the optimal control input from Eq. (18) into the Hamiltonian (17) while maintaining $H[V, u_V^*, W_V^*(V)] = 0$ gives the HJB equation and the necessary and sufficient condition for optimal control to be [18]

$$0 = Q(V) + W_V^{*T}(V)f(V) - \frac{W_V^{*T}(V)gB^{-1}g^T W_V^*(V)}{4} \quad (19)$$

with $W^*(0) = 0$. At this point, Lemma 1 must be introduced.

Lemma 1 Boundedness of System States

Given the unmanned helicopter system (15) with cost function (16) and optimal control (18), let $J_1(V)$ be a continuously differentiable, radially unbounded Lyapunov candidate function such that $\dot{J}_1(V) = J_{1V}^T(V)\dot{V} = J_{1V}^T(V)[f(V) + gu_V^*] < 0$ with $J_{1V}(V)$ the partial derivative of $J_1(V)$. In addition, let $\bar{Q}(V) \in \mathbb{R}^{6 \times 6}$ be a positive-definite matrix satisfying $\|\bar{Q}(V)\| = 0$ only if $\|V\| = 0$ and $\bar{Q}_{\min} \leq \|\bar{Q}(V)\| \leq \bar{Q}_{\max}$ for $V_{\min} \leq \|V\| \leq V_{\max}$ for positive constants \bar{Q}_{\min} , \bar{Q}_{\max} , V_{\min} , and V_{\max} . Also, let $Q(V)$ satisfy

$$\lim_{V \rightarrow \infty} \bar{Q}(V) = \infty$$

as well as

$$W_V^{*T}\bar{Q}(V)J_{1V} = r(V, u_V^*) = Q(V) + u_V^{*T}Bu_V^* \quad (20)$$

then the following relation is true:

$$J_{1V}^T[f(V) + gu_V^*] = -J_{1V}^T\bar{Q}(V)J_{1V} \quad (21)$$

D. SINGLE-ONLINE-APPROXIMATOR-BASED OPTIMAL CONTROL OF HELICOPTER

Usually, in adaptive critic-based techniques, two OLAs [15] are used for optimal control, with one used to approximate the cost function while the other is used to generate the control action. In this chapter, the adaptive critic for optimal control of a helicopter is realized online using only one OLA. For the SOLA to learn the cost function, the cost function is rewritten using the OLA representation as

$$W(V) = \Gamma^T\Phi(V) + \varepsilon(V) \quad (22)$$

where $\Gamma \in \mathbb{R}^L$ is the constant target OLA vector, $\Phi(V)$ is a linearly independent basis vector that satisfies $\Phi(V) = 0$, and $\varepsilon(V)$ is the OLA reconstruction error. The basis vector used in this case is $\Phi(V) = [V \ V^2 \ V^3 \ \sin(V) \ \sin(2V) \ \tanh(V) \ 1]^T$. The target OLA vector and reconstruction errors are assumed to be upper bounded according to $\|\Gamma\| \leq \Gamma_M$ and $\|\varepsilon(V)\| \leq \varepsilon_M$, respectively [19], which is true for any real helicopter and for any physically realizable trajectory. The gradient of the OLA cost function in Eq. (22) is written as

$$\frac{\partial W(V)}{\partial V} = W_V(V) = \nabla_V^T \Phi(V) \Gamma + \nabla_V \varepsilon(V) \quad (23)$$

Using Eq. (23), the optimal control input in Eq. (18) and the HJB equation (19) can be written as

$$u_V^* = \frac{-B^{-1}g^T \nabla_V^T \Phi(V) \Gamma}{2} - \frac{B^{-1}g^T \nabla_V \varepsilon(V)}{2} \quad (24)$$

$$H^*(V, \Gamma) = Q(V) + \Gamma^T \nabla_V \Phi(V) f(V) - \frac{\Gamma^T \nabla_V \Phi(V) C \nabla_V^T \Phi(V) \Gamma}{4} + \varepsilon_{\text{HJB}} = 0 \quad (25)$$

where $C = gB^{-1}g^T > 0$ is bounded such that $C_{\min} \leq \|C(V)\| \leq C_{\max}$ for known constants C_{\min} and C_{\max} and

$$\begin{aligned} \varepsilon_{\text{HJB}} &= \nabla_V \varepsilon^T \left\{ f(V) - \frac{1}{2} g B^{-1} g^T [\nabla_V^T \Phi(V) \Gamma + \nabla_V \varepsilon] \right\} + \frac{1}{4} \nabla_V \varepsilon^T g B^{-1} g^T \nabla_V \varepsilon \\ &= \nabla_V \varepsilon^T [f(V) + g u_V^*] + \frac{1}{4} \nabla_V \varepsilon^T C \nabla_V \varepsilon \end{aligned}$$

is the OLA reconstruction error. The OLA estimate of Eq. (22) is

$$\hat{W}(V) = \hat{\Gamma}^T \Phi(V) \quad (26)$$

with $\hat{\Gamma}$ the OLA estimate of the target vector Γ . In the same way, the estimate for the optimal control input in Eq. (24) and the approximate Hamiltonian in Eq. (25) in terms of $\hat{\Gamma}$ can be expressed as

$$\hat{u}_V^* = \frac{-B^{-1}g^T(V) \nabla_V^T \Phi(V) \hat{\Gamma}}{2} \quad (27)$$

In this work, an initial stabilizing control is not required to implement the proposed SOLA-based scheme. Moreover, Lyapunov analysis shows that the estimated control inputs approach the optimal control inputs with a bounded error. The approximate Hamiltonian can now be written as

$$\hat{H}^*(V, \hat{\Gamma}) = Q(V) + \hat{\Gamma}^T \nabla_V \Phi(V) f(V) - \frac{\hat{\Gamma}^T \nabla_V \Phi(V) C \nabla_V^T \Phi(V) \hat{\Gamma}}{4} \quad (28)$$

Considering the definition of the OLA approximation of the cost function (26) and the Hamiltonian function (28), it is clear that both converge to zero when $\|V\| = 0$. Consequently, once the system states have converged to zero, the cost function approximation can no longer be updated, which means that there needs to be a persistency of excitation (PE) requirement for the inputs to the cost function OLA [19]. The system states must be persistently excited long enough for the OLA to learn the optimal cost function. Recollecting the HJB equation in Eq. (25), the OLA estimate $\hat{\Gamma}$ should be tuned to minimize $\hat{H}^*(V, \hat{\Gamma})$. However, merely tuning $\hat{\Gamma}$ to minimize $\hat{H}^*(V, \hat{\Gamma})$ does not ensure the stability of the nonlinear helicopter system during the OLA learning process.

Therefore, the OLA tuning algorithm is designed to minimize Eq. (28) while considering the stability of Eq. (15) and is given here as

$$\begin{aligned} \dot{\hat{\Gamma}} = & -\alpha_1 \frac{\hat{\beta}}{(\hat{\beta}^T \hat{\beta} + 1)^2} \left[Q(V) + \hat{\Gamma}^T \nabla_V \Phi(V) f(V) - \frac{\hat{\Gamma}^T \nabla_V \Phi(V) C \nabla_V^T \Phi(V) \hat{\Gamma}}{4} \right] \\ & + \Sigma(V, \hat{u}_V) \frac{\alpha_2}{2} \nabla_V \Phi(V) g B^{-1} g^T J_{1V}(V) \end{aligned} \quad (29)$$

where $\hat{\beta} = \nabla_V \Phi(V) f(V) - \nabla_V \Phi(V) C \nabla_V^T \Phi(V) \hat{\Gamma} / 2$, $\alpha_1 > 0$, and $\alpha_2 > 0$ are design constants; $J_{1V}(V)$ is as in Lemma 1; and the operator $\Sigma(V, \hat{u}_V)$ is given by

$$\Sigma(V, \hat{u}_V) = \begin{cases} 0 & \text{if } J_{1V}^T(V) \dot{V} = J_{1V}^T(V) \left[\frac{f(V) - g B^{-1} g^T \nabla_V^T \Phi(V) \hat{\Gamma}}{2} \right] < 0 \\ 1 & \text{otherwise} \end{cases} \quad (30)$$

The first term in Eq. (29) is the portion of the tuning law that tries to minimize Eq. (28) and has been derived using a normalized gradient descent scheme with the auxiliary HJB error defined as shown here:

$$E_{\text{HJB}} = \frac{[\hat{H}^*(V, \hat{\Gamma})]^2}{2} \quad (31)$$

The second term in the OLA tuning law in Eq. (29) ensures that the system states remain bounded while the SOLA scheme learns the optimal cost function.

The dynamics of the OLA parameter estimation error is considered as $\tilde{\Gamma} = \Gamma - \hat{\Gamma}$. Because this yields $Q(V) = -\Gamma^T \nabla_V \Phi(V) f(V) + \Gamma^T \nabla_V \Phi(V) \times C \nabla_V^T \Phi(V) \Gamma / 4 - \varepsilon_{\text{HJB}}$ from Eq. (25), the approximate HJB equation in Eq. (25) can be expressed in terms of $\tilde{\Gamma}$ as

$$\begin{aligned} \hat{H}(V, \hat{\Gamma}) = & -\tilde{\Gamma}^T \nabla_V \Phi(V) f(V) + \frac{1}{2} \tilde{\Gamma}^T \nabla_V \Phi(V) C \nabla_V^T \Phi(V) \Gamma \\ & - \frac{1}{4} \tilde{\Gamma}^T \nabla_V \Phi(V) C \nabla_V^T \Phi(V) \tilde{\Gamma} - \varepsilon_{\text{HJB}} \end{aligned} \quad (32)$$

Then, because $\dot{\tilde{\Gamma}} = \dot{\hat{\Gamma}}$ and $\hat{\beta} = \nabla_V \Phi(V)(\dot{V}^* + C\nabla_V \varepsilon/2) + \nabla_V \Phi(V)C\nabla_V^T \times \Phi(V)\tilde{\Gamma}/2$, where $\dot{V} = f(V) + gu_V^*$, the error dynamics of Eq. (29) are

$$\begin{aligned} \dot{\tilde{\Gamma}} = & \frac{\alpha_1}{\rho_1^2} \left[\nabla_V \Phi(V) \left(\dot{V}_1^* + \frac{C\nabla_V \varepsilon}{2} \right) + \frac{\nabla_V \Phi(V)C\nabla_V^T \Phi(V)\tilde{\Gamma}}{2} \right] \\ & \times \left[\tilde{\Gamma}^T \nabla_V \Phi(V) \left(\dot{V}_1^* + \frac{C\nabla_V \varepsilon}{2} \right) + \frac{\tilde{\Gamma}^T \nabla_V \Phi(V)C\nabla_V^T \Phi(V)\tilde{\Gamma}}{2} + \varepsilon_{\text{HJB}} \right] \\ & - \Sigma(V, \hat{u}_V) \frac{\alpha_2}{2} \nabla_V \Phi(V) g B^{-1} g^T J_{1V}(V) \end{aligned} \quad (33)$$

where $\rho_1 = (\hat{\beta}^T \hat{\beta} + 1)$. Next, it is necessary to examine the stability of the SOLA-based adaptive scheme for optimal control along with the stability of the helicopter system.

Definition: An equilibrium point V_e is said to be uniformly ultimately bounded (UUB) if there exists a compact set $S \subset \mathbb{R}^4$ such that for every $V_0 \in S$ there exists a bound D and time $T(D, V_0)$ such that $\|V(t) - V_e\| \leq D$ for all $t \geq t_0 + T$.

Theorem 2 SOLA-Based Scheme for Convergence to the HJB Function and System Stability

Given the unmanned helicopter system in Eq. (5) with target HJB equation (17), let the tuning law for the SOLA be given by Eq. (29). Then there exist constants b_{JV} and b_Γ such that the OLA approximation error $\tilde{\Gamma}$ and $\|J_{1V}(V)\|$ are UUB for all $t \geq t_0 + T$ with ultimate bounds given by $\|J_{1V}(V)\| \leq b_{JV}$ and $\|\tilde{\Gamma}\| \leq b_\Gamma$. Further, OLA reconstruction error $\|W^* - \tilde{W}\| \leq \varepsilon_{r1}$ and $\|u_V^* - \hat{u}_V\| \leq \varepsilon_{r2}$ for small positive constants ε_{r1} and ε_{r2} , respectively. Proof will be provided later for the tracking case.

E. NN CONTROL SCHEME FOR TRACKING CONTROL

In this section, based on the information provided by the kinematic controller, the optimal control input u_v^* is designed to ensure that the unmanned helicopter system tracks a desired trajectory in an optimal manner. For optimal tracking, the desired trajectory is defined as

$$\dot{V}_d = f(V_d) + gu_v^* \quad (34)$$

where $f(V_d)$ is the internal dynamics of the helicopter system rewritten in terms of the desired state, g remains as defined earlier, and u_v^* is the desired control input corresponding to the desired states. Next, the state tracking error is defined as

$$e = V - V_d \quad (35)$$

and considering Eqs. (15) and (34), the tracking error dynamics in Eq. (35) can be written as

$$\dot{e} = f(V) + gu_V + \dot{V}_d = f_e(e) + gu_e \quad (36)$$

where $f_e(e) = f(V) - f(V_d)$ and $u_e = u_v - u_v^*$. To control Eq. (34) in an optimal manner, the control policy u_e should be selected such that it minimizes the cost function given by

$$W_T[e(t)] = \int_t^\infty r[e(\tau), u_e(\tau)] d\tau \quad (37)$$

where $r[e(\tau), u_e(\tau)]$ is defined in a similar manner to $r[V(\tau), u_v(\tau)]$ with $V(\tau)$ and $u_v(\tau)$ replaced with $e(\tau)$ and $u_e(\tau)$, respectively.

After this, the Hamiltonian for the HJB tracking problem is defined as

$$H_T(e, u) = r(e, u) + W_{Te}^T(e)[f_e(e) + gu_e] \quad (38)$$

where $W_{Te}(e)$ is the gradient of $W_T(e)$ with respect to e . The basis function used for the neural network law is $\Phi(e) = [\nabla e \nabla e^2 \nabla e^3 \nabla \sin(e) \nabla \sin(2e) \nabla \tanh(e) 1]$, with system errors bounded such that $\|\Phi(e)\| \leq \Phi_M$ and $\|\nabla_e \Phi(e)\| \leq \Phi'_M$, which is true for any real helicopter and for any physically realizable trajectory. Now applying stationarity condition $\partial H(e, u_e)/\partial u_e = 0$, the optimal control input is found to be

$$u_e^*(e) = \frac{-B^{-1}g^T W_{Te}^*(e)}{2} \quad (39)$$

with B as defined in Sec. III.C, and $u_e^*(e) \in \mathbb{R}^{4 \times 1}$. Substituting the optimal control input from Eq. (39) into the Hamiltonian (38) generates the HJB equation for the tracking problem as

$$0 = Q_e(e) + W_{Te}^{*T}(e)f_e(e) - \frac{W_{Te}^{*T}(e)gB^{-1}g^T W_{Te}^*(e)}{4} \quad (40)$$

with $W_T^*(0) = 0$ and Q_e defined similarly to $Q(V)$ in Sec. III.C. Next, it is apparent that the expression for the optimally augmented control input in Eq. (39) can be written as

$$\hat{u}_v = u_d - \frac{B^{-1}g^T W_{Te}^*(e)}{2} \quad (41)$$

where the desired feedforward control input is as presented earlier. Note that this u_v^* becomes the input U , which is used as the system input. The feedforward control input is used for hovering and trajectory tracking and also for elimination of the need for the $f(0) = 0$ optimal controller requirement, which would otherwise be in place for both regulation and trajectory tracking for the system.

The proofs to be introduced shortly will be built on the basis of the work of [15] and [17]. It is found that the control input consists of a predetermined feedforward term u_d and an optimal feedback term that is a function of the gradient of the optimal cost function. To implement the optimal control in Eq. (39), the SOLA-based control law in Sec. III.D is used to learn the optimal feedback tracking control after necessary modifications, such that the OLA tuning algorithm is able to minimize the Hamiltonian while maintaining the stability of the helicopter system.

Lemma 1 has been introduced already and gives the boundedness of $\|J_{1V}\|$ and therefore the system states, which is necessary for Theorem 2. Theorem 2 was also introduced already and reveals that the SOLA convergence to the HJB function is UUB for regulation of the states. Theorem 3, which will be provided next, establishes the optimality of the single network adaptive critic controller feedback term. Lemma 4 will then be provided because it provides a stability condition needed for the proof for Theorem 5. Theorem 5 gives results similar to Theorem 2, except that it is performed in terms of the state error and is used to demonstrate tracking convergence. In addition, Theorem 5 includes the feedforward term stability and the stability of the entire resulting system.

Theorem 3 Optimality and Convergence of the Single Network Adaptive Critic Controller Feedback Term

Given the nonlinear helicopter UAV system defined in Eqs. (4) and (5), with target HJB equation (19), let the SOLA tuning law be given by Eq. (29). Let the feedforward control input be given by Eq. (14). Then the velocity tracking error and NN parameter estimation errors of the cost function term are UUB for all $t \geq t_0 + T$, and the tracking error feedback system is controlled in a near-optimal manner. That is, $\|u_e^* - \hat{u}_e\| \leq \varepsilon_u$ for a small positive constant ε_u .

The proof is performed by restating Theorem 1 in terms of state error e in order to begin the stability analysis of the optimal controller. But first, the Lyapunov candidate function and its derivative are provided:

$$J_{\text{HJB}} = \alpha_4 J_2(e) + \frac{\tilde{\Gamma}^T \tilde{\Gamma}}{2}$$

$$\dot{J}_{\text{HJB}} = \alpha_4 J_{e2}^T(e) \dot{e} + \tilde{\Gamma}^T \dot{\tilde{\Gamma}}$$

Theorems 3 and 5 are proven identically the same way as Theorem 2, with proof to follow shortly for Theorem 5.

For the proof of Theorem 2, Lemma 4 will be needed.

Lemma 4 Stability Condition

If an affine nonlinear system is asymptotically stable and has an L_2 gain below a bound γ , and if the cost function given in [15] is smooth, then the closed-loop dynamics are asymptotically stable.

Theorem 5 Overall System Stability

Given the unmanned helicopter system in Eq. (5) with target HJB equation (19), let the tuning law for the SOLA be given by Eq. (29), and let the feedforward control input be as in Eq. (14). Then there exist constants b_{J_e} and $b_{\tilde{\Gamma}}$ such that the OLA approximation error $\tilde{\Gamma}$ and $\|J_{1e}(e)\|$ are UUB for all $t \geq t_0 + T$ with ultimate bounds given by $\|J_{1e}(e)\| \leq b_{J_e}$ and $\|\tilde{\Gamma}\| \leq b_{\tilde{\Gamma}}$. Further, OLA reconstruction error $\|W^* - \hat{W}\| \leq \varepsilon_{r1}$ and $\|u_e^* - \hat{u}_e\| \leq \varepsilon_{r2}$ for small positive constants ε_{r1} and ε_{r2} .

Proof: First, begin with the positive-definite Lyapunov function candidate

$$J_{\text{HJB}} = \alpha_2 J_1(e) + \frac{\tilde{\Gamma}^T \tilde{\Gamma}}{2}$$

Differentiating, one obtains

$$\dot{J}_{\text{HJB}} = \alpha_2 J_{1e}^T(e) \dot{e} + \tilde{\Gamma}^T \dot{\tilde{\Gamma}}$$

with $J_1(e)$ and $J_{1e}(e)$. If $\|e\| = 0$, $J_{\text{HJB}}(e) = \tilde{\Gamma}^T \tilde{\Gamma}/2$, $\dot{J}_{\text{HJB}}(e) = 0$, and $\|\tilde{\Gamma}\|$ remains a bounded constant. For online learning, however, it is the case that $\|e\| > 0$. Then, using the affine nonlinear system, the optimal control input, and the tuning law's error dynamics along with the derivative of the Lyapunov candidate function J_{HJB} , one arrives at

$$\begin{aligned} \dot{J}_{\text{HJB}} = & \alpha_2 J_{1e}^T(e) \left[f_e(e) - \frac{1}{2} g B^{-1} g^T \nabla_e^T \Phi(e) \hat{\Gamma} \right] - \frac{\alpha_1}{\rho^2} \left[\tilde{\Gamma}^T \nabla_e \Phi_e \left(\dot{e}_1^* + \frac{C \nabla_e \mathcal{E}}{2} \right) \right]^2 \\ & - \frac{\alpha_1}{8\rho^2} [\tilde{\Gamma}^T \nabla_e \Phi_e C \nabla_e^T \Phi(e) \tilde{\Gamma}]^2 \\ & - \frac{3\alpha_1}{4\rho^2} \tilde{\Gamma}^T \nabla_e \Phi(e) \left(\dot{e}_1^* + \frac{C \nabla_e \mathcal{E}}{2} \right) \tilde{\Gamma}^T \nabla_e \Phi(e) C \nabla_e^T \Phi(e) \tilde{\Gamma} \\ & - \frac{\alpha_1}{\rho^2} \tilde{\Gamma}^T \nabla_e \Phi(e) \left(\dot{e}_1^* + \frac{C \nabla_e \mathcal{E}}{2} \right) \mathcal{E}_{\text{HJB}} \\ & - \frac{\alpha_1}{2\rho^2} \tilde{\Gamma}^T \nabla_e \Phi(e) C \nabla_e^T \Phi(e) \tilde{\Gamma} \mathcal{E}_{\text{HJB}} - \Sigma(e, \hat{u}_e) \frac{\alpha_2}{2} \tilde{\Gamma}^T \nabla_e \Phi(e) g B^{-1} g^T J_{1e}^T(e) \end{aligned}$$

Completing the square, simplifying, and using the Cauchy–Schwartz inequality yields

$$\begin{aligned} \dot{J}_{\text{HJB}} \leq & \alpha_2 J_{1e}^T(e) \left[f_e(e) - \frac{1}{2} g B^{-1} g^T \nabla_e^T \Phi(e) \hat{\Gamma} \right] \\ & - \Sigma(e, \hat{u}_e) \frac{\alpha_2}{2} \tilde{\Gamma}^T \nabla_e \Phi(e) g B^{-1} g^T J_{1e}^T(e) - \frac{\alpha_1}{\rho^2} \|\tilde{\Gamma}\|^4 \beta_1 \\ & + \frac{\alpha_1}{\rho^2} \eta(\varepsilon) + \frac{\alpha_1}{\rho^2} \beta_2 \delta^4(e) \end{aligned}$$

where

$$\beta_1 = \frac{\nabla \Phi_{\min}^4 C_{\min}^2}{64} \quad \beta_2 = \frac{1024}{C_{\min}^2} + \frac{3}{2} \quad \eta(\varepsilon) = \frac{64}{C_{\min}^2} + \frac{3(\varepsilon_M'^4 + \varepsilon_M'^4 C_{\max}^2)}{2}$$

ε_M' is an upper bound on the OLA reconstruction error, and $0 < \nabla \Phi_{\min} \leq \|\nabla \Phi(e)\|$. Now it is necessary to consider the case $\Sigma(e, \hat{u}_e) = 0$:

$$\dot{J}_{\text{HJB}} \leq -(\alpha_2 \dot{e}_{\min} - \alpha_1 \beta_2 K^*) \|J_{1e}(e)\| - \frac{\alpha_1 \|\tilde{\Gamma}\|^4 \beta_1}{\rho^2} + \frac{\alpha_1 \eta(\varepsilon)}{\rho^2}$$

This is less than zero if

$$\frac{\alpha_2}{\alpha_1} > \frac{\beta_2 K^*}{\dot{\epsilon}_{\min}}, \quad \|J_{1e}(e)\| > \frac{\alpha_1 \eta(\epsilon)}{(\alpha_2 \dot{\epsilon}_{\min} - \alpha_1 \beta_2 K^*)} \equiv b_{Je0},$$

or

$$\|\tilde{\Gamma}\| > \sqrt[4]{\frac{\eta(\epsilon)}{\beta_1}} \equiv b_{\Gamma 0}$$

Next, consider the case $\Sigma(e, \hat{u}_e) = 1$:

$$\begin{aligned} j_{\text{HJB}} &\leq \alpha_2 J_{1e}^T(e) \left\{ f_e(e) - \frac{1}{2} C [\nabla_e^T \Phi(e) \Gamma + \nabla_e \varepsilon] \right\} + \frac{\alpha_2}{2} J_{1e}^T(e) C \nabla_e \varepsilon \\ &\quad - \frac{\alpha_1}{\rho^2} \|\tilde{\Gamma}^T\|^4 \beta_1 + \frac{\alpha_1 \eta(\epsilon)}{\rho^2} + \frac{\alpha_1}{\rho^2} \beta_2 \delta^4(e) \\ &= \alpha_2 J_{1e}^T(e) [f_e(e) + g u^*] + \frac{\alpha_2}{2} J_{1e}^T(e) C \nabla_e \varepsilon \\ &\quad - \alpha_1 \frac{\alpha_1 \beta_1}{\rho^2} \|\tilde{\Gamma}\|^4 + \frac{\alpha_1}{\rho^2} \beta_2 K^* \|J_{1e}\| \end{aligned}$$

Lemma 4 yields

$$j_{\text{HJB}} \leq -\frac{\alpha_2 \bar{Q}_{e,\min} \|J_{1e}(e)\|^2}{2} - \frac{\alpha_1 \|\tilde{\Gamma}\|^4 \beta_1}{\rho^2} + \frac{\alpha_1 \eta(\epsilon)}{\rho^2} + \frac{\alpha_2 C_{\max}^2 \varepsilon_M'^2}{(4\bar{Q}_{e,\min})} + \frac{\alpha_1^2 \beta_2^2 K^{*2}}{(\alpha_2 \rho^4 \bar{Q}_{e,\min})}$$

with $0 < \bar{Q}_{e,\min} \leq \|Q_e(e)\|$. Lemma 1 and Lemma 4 will then ensure $j_{\text{HJB}} < 0$ given that

$$\|J_{1e}(e)\| > \sqrt{\frac{C_{\max}^2 \varepsilon_M'^2}{(2\bar{Q}_{e,\min}^2)}} \equiv b_{Je1'} \quad (42)$$

and

$$\|\tilde{\Gamma}\| > \sqrt[4]{\frac{\eta(\epsilon)}{\beta_1} + \frac{\alpha_1 \beta_2^2 K^{*2}}{(\beta_1 \alpha_2 \bar{Q}_{e,\min})}} \equiv b_{\Gamma 1} \quad (43)$$

which allows the conclusion that $\|W^*(e) - \hat{W}(e)\| \leq \|\tilde{\Gamma}\| \|\Phi(e)\| + \varepsilon_M \leq b_{\Gamma} \Phi_M + \varepsilon_M \equiv \varepsilon_{r1}$ and $\|u_e^*(e) - \hat{u}_e(e)\| \leq \lambda_{\max}(B^{-1}) g_M b_{\Gamma} \Phi_M' / 2 + \lambda_{\max}(B^{-1}) g_M \varepsilon_M' / 2 \equiv \varepsilon_{r2}$. At this point, it is necessary to go back and provide proof for Lemma 1. Applying the optimal control input to an affine nonlinear system, the cost function becomes

$$\dot{W}^*(e) = W_e^{*T}(e) \dot{e} = W_e^{*T}(e) [f_e(e) + g u_e^*] = -Q_e(e) - u_e^{*T} B u_e^*$$

Because

$$W_e^{*T} \bar{Q}_e(e) J_{1e} = r(e, u_e^*) = Q_e(e) + u_e^{*T} B u_e^*$$

one may obtain

$$\begin{aligned} [f_e(e) + g u_e^*] &= -(W_e^* W_e^{*T})^{-1} W_e^* [Q_e(e) + u_e^{*T} B u_e^*] \\ &= -(W_e^* W_e^{*T})^{-1} W_e^* W_e^{*T} \bar{Q}_e(e) J_{1e} \\ &= -\bar{Q}_e(e) J_{1e} \end{aligned}$$

from which one then has

$$J_{1e}^T [f_e(e) + g u_e^*] = -J_{1e}^T \bar{Q}_e(e) J_{1e}$$

concluding the proof for Lemma 1.

Next, the feedforward control will be considered. The Lyapunov candidate function for the feedforward control is

$$J_{\text{feedforward}} = S_1 + S_2 + S_3 + S_4$$

with $S_1 = \frac{1}{2} \delta_1^T \delta_1$, $S_2 = \frac{1}{2} \delta_2^T \delta_2$, $S_3 = \frac{1}{2} \delta_3^T \delta_3 + \frac{1}{2} \epsilon_3^T \epsilon_3$, and $S_4 = \frac{1}{2} \delta_4^T \delta_4 + \frac{1}{2} \epsilon_4^T \epsilon_4$. It has been shown that this selection of Lyapunov candidates will guarantee stability in [17]. Applying elements integral to Eq. (14) gives the derivative of the Lyapunov function

$$\dot{J}_{\text{feedforward}} = \dot{S}_1 + \dot{S}_2 + \dot{S}_3 + \dot{S}_4 = -\frac{1}{m} \delta_1^T \delta_1 - \delta_2^T \delta_2 - \delta_3^T \delta_3 - \delta_4^T \delta_4 - \epsilon_3^T \epsilon_3 - \epsilon_4^T \epsilon_4$$

so $\dot{J}_{\text{feedforward}} < 0$. To quickly review the elements in this stability analysis, δ_1 is used for the error in the tracking along with ϵ_3 , δ_2 regulates the translational velocity, δ_3 and δ_4 take roll and pitch angles into consideration, and ϵ_3 and ϵ_4 are used for the error in the orientation (yaw) and corresponding rotational velocity. The stability of the entire system is now considered. Combining the proofs for optimal and feedforward controller stability by taking $J = J_{\text{HJB}} + J_{\text{feedforward}}$,

$$\begin{aligned} \dot{J}_{\text{HJB}} + \dot{J}_{\text{feedforward}} &= -\frac{\alpha_2 \bar{Q}_{e,\min} \|J_{1e}(e)\|^2}{2} - \frac{\alpha_1 \|\tilde{\Gamma}\|^4 \beta_1}{\rho^2} + \frac{\alpha_1 \eta(\varepsilon)}{\rho^2} \\ &\quad + \frac{\alpha_2 C_{\max}^2 \varepsilon_M'^2}{(4\bar{Q}_{e,\min})} + \frac{\alpha_1^2 \beta_2^2 K^{*2}}{(\alpha_2 \rho^4 \bar{Q}_{e,\min})} \\ &\quad - \frac{1}{m} \delta_1^T \delta_1 - \delta_2^T \delta_2 - \delta_3^T \delta_3 - \delta_4^T \delta_4 - \epsilon_3^T \epsilon_3 - \epsilon_4^T \epsilon_4 \end{aligned}$$

Then $\dot{J}_{\text{HJB}} + \dot{J}_{\text{feedforward}} < 0$ provided that Eqs. (42) and (43) hold. In other words, the overall system is UUB with the bounds from Eqs. (42) and (43), completing the proof.

The dynamics presented at the beginning of the chapter provide $\bar{S}(\omega) = [0^{3 \times 1}, -\omega \times J\omega]^T$. Actually, however, this is a simplification of the real dynamics, which include an additional coupling term such that $\bar{S}(\omega) = [R(\Theta)K\omega_d, -\omega \times J\omega]^T$, with

$$K = \frac{1}{I_M} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & \frac{1}{l_t} \\ 0 & 0 & 0 \end{bmatrix}$$

This coupling term is relatively small, but the robustness against neglecting the term has been demonstrated and is available for the interested reader in [17].

IV. SIMULATION RESULTS

Simulation results for the unmanned helicopter are presented in this section using the model presented in Sec. II. All simulations are performed in Simulink and demonstrate the performance of the proposed control scheme when the helicopter is hovering, landing, and tracking trajectories. The simulations take into account the aerodynamic features presented as part of the helicopter model earlier in this chapter, but do not include hardware-specific sensor noise or actuator constraints. The constants used for simulation are $\bar{g} = 9.8 \text{ m/s}^2$, $m = 9.6 \text{ kg}$, $\mathcal{J} = \text{diag}([0.40 \ 0.56 \ 0.29]^T)$, $\text{kg} \cdot \text{m}^2$, $p = \text{diag}([1.1 \ 1.1 \ 1.1]^T)$, $\alpha_1 = 100$, $\alpha_2 = 1$, $l_t = 1.2 \text{ m}$, $l_m = 0.27 \text{ m}$, $Q_M = 0.002$, and $Q_T = 0.0002$. The optimal controller used seven hidden layer neurons for all simulations in this section, with gains $B = \text{diag}([0.1 \ 0.1 \ 0.1 \ 0.001]^T)$. The NN basis function is as given earlier. All NN weights are initialized to zero. The following cases were considered:

Case I—Regulation:

$$[x_d \ y_d \ z_d]^T = [15 \ 25 \ 50]^T \quad (44)$$

An additional plot after Case I shows the helicopter hovering in a circle about a fixed point.

Case II—Tracking:

$$\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} = \begin{bmatrix} 10 \sin(0.025t) \\ 5 \sin(0.025t) \\ \sin(0.01t) \end{bmatrix} \quad (45)$$

For both cases, the heading (yaw) was set to zero. For the two final plots in Figs. 6 and 7, the heading (yaw) is changed midmaneuver from 1 to -1 radian. This is clearly visible in Fig. 6.

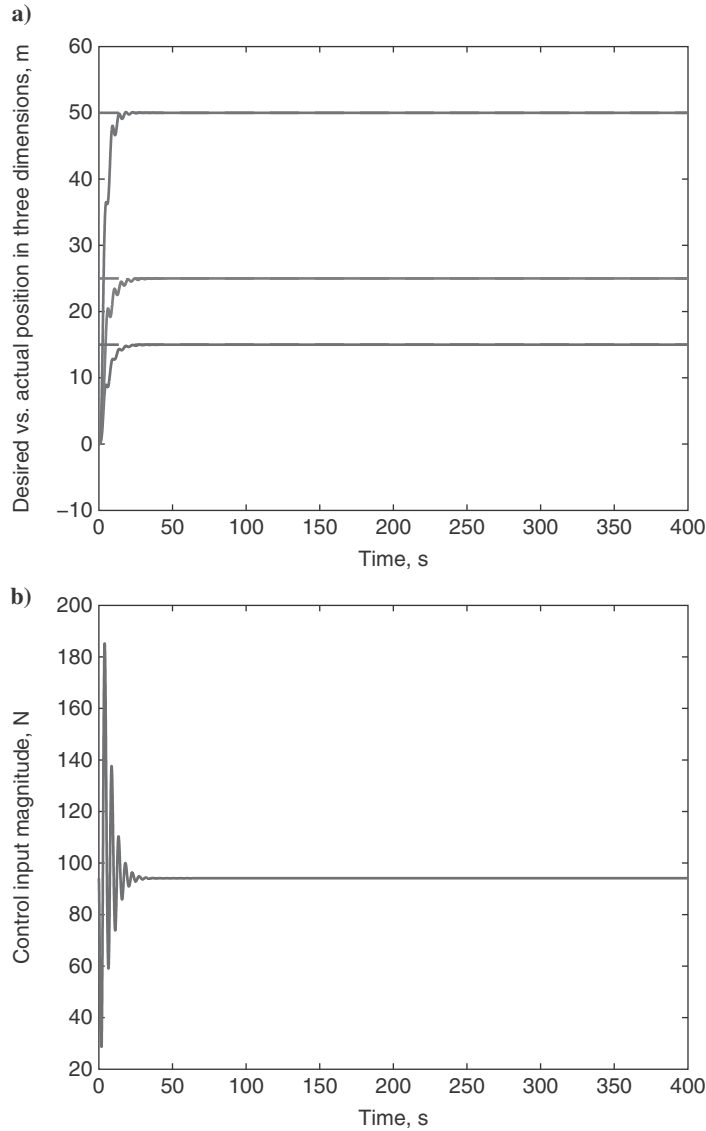


FIG. 3 Helicopter regulating its ability in three dimensions: a) helicopter position vs time for regulation and b) feedforward main rotor thrust control input vs time for regulation.

Figure 3 shows the helicopter’s ability to regulate its position in three dimensions. Clearly, this capability is excellent. Figure 3 also gives the backstepping controller feedforward control input applied to generate force for regulation. Figure 4

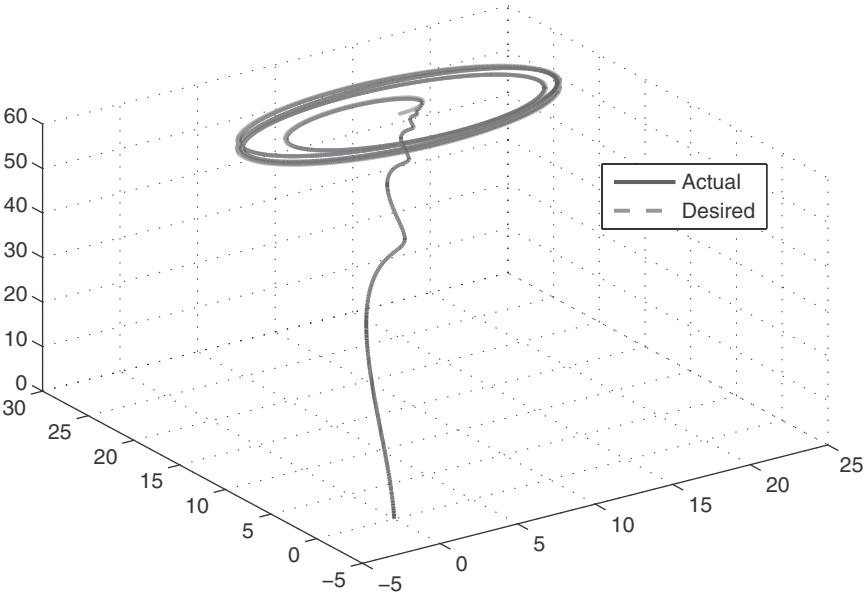


FIG. 4 Three-dimensional perspective of position during takeoff and circular maneuver.

demonstrates the helicopter’s ability to follow a trajectory in two dimensions while hovering. Figure 5 shows the helicopter’s ability to track a trajectory in three dimensions. Figure 6 provides a three-dimensional view of the helicopter performing a landing maneuver, showing both the position and orientation as the

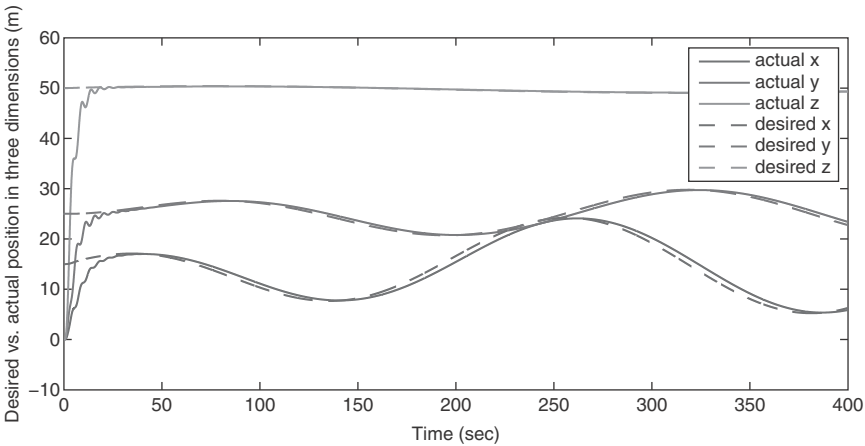


FIG. 5 Helicopter position vs time for tracking.

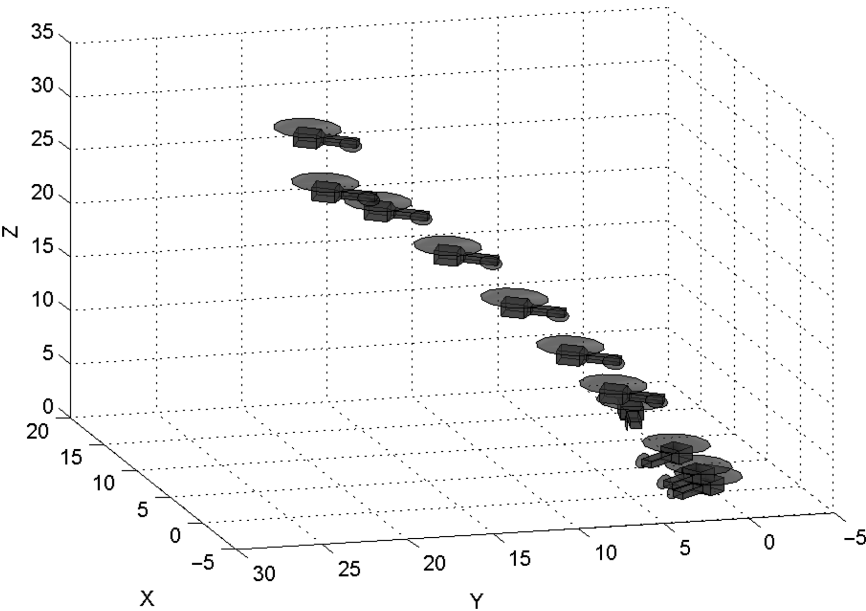


FIG. 6 Three-dimensional perspective of position and orientation during landing.

helicopter lands. It is clear that the helicopter’s heading may be altered by the controller as desired during a maneuver. Figure 7 provides a three-dimensional view of the helicopter performing a landing maneuver, but in this case shows the actual vs desired trajectories throughout the maneuver. Although not shown here, simulation results also confirm that the NN weights remain bounded.

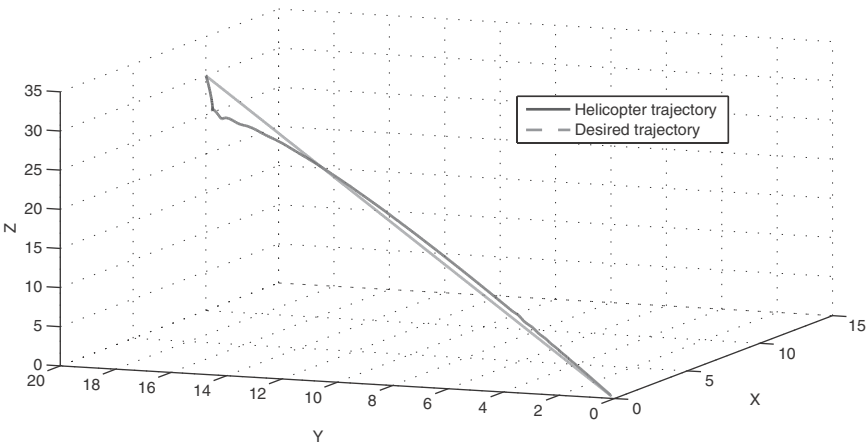


FIG. 7 Three-dimensional perspective of position during landing.

V. CONCLUSION

A neural-network-based optimal control law has been proposed, which uses a single online approximator for optimal regulation and tracking control of an unmanned helicopter with known dynamics having a dynamic model in strict-feedback form. The SOLA-based adaptive approach is designed to learn the infinite horizon continuous-time HJB equation, and the corresponding optimal control input that minimizes the HJB equation is calculated forward in time. A feedforward controller has been introduced to compensate for the helicopter's weight and requirement for rotor thrust when in hover and to permit trajectory tracking. Further, it has been shown that the estimated control input approaches the target optimal control input with a small bounded error. A kinematic control structure was used to obtain the desired velocity such that the desired position is achieved. The stability of the system was analyzed, and simulation results confirm that the unmanned helicopter is capable of regulation and trajectory tracking.

ACKNOWLEDGMENTS

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0077. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES

- [1] Koo, T. J., and Sastry, S., "Output Tracking Control Design of a Helicopter Model Based on Approximate Linearization," *Proceedings of the 37th IEEE Conference on Decision and Control*, Vol. 4, IEEE, Piscataway, NJ, 1998, pp. 3635–3640.
- [2] Mettler, B. F., Tischler, M. B., and Kanade, T., "System Identification Modelling of a Small-Scale Rotorcraft for Flight Control Design," *International Journal of Robotics Research*, Vol. 20, No. 10, 2000, pp. 795–807.
- [3] Hovakimyan, N., Kim, N., Calise, A. J., and Prasad, J. V. R., "Adaptive Output Feedback for High-Bandwidth Control of an Unmanned Helicopter," *Proceedings of AIAA Guidance, Navigation, and Control Conference*, Vol. 25, AIAA, Reston, VA, 2001, pp. 1–11.
- [4] Johnson, E. N., and Kannan, S. K., "Adaptive Trajectory Control for Autonomous Helicopters," *Journal of Guidance, Control and Dynamics*, Vol. 28, No. 3, 2005, pp. 524–538.
- [5] Palunko, I., Fierro, R., and Sultan, C., "Nonlinear Modeling and Output Feedback Control Design for a Small-Scale Helicopter," *Proceedings of 17th Mediterranean Conference on Control and Automation*, IEEE, Piscataway, NJ, 2009, pp. 1251–1256.

- [6] Ahmed, B., Pota, H. R., and Garratt, M., "Flight Control of a Rotary Wing UAV Using Backstepping," *International Journal of Robust and Nonlinear Control*, Vol. 20, No. 6, 2010, pp. 639–658.
- [7] Enns, R., and Si, J., "Helicopter Trimming and Tracking Control Using Direct Neural Dynamic Programming," *IEEE Transactions on Neural Networks*, Vol. 14, No. 4, 2003, pp. 929–939.
- [8] Lee, S., Ha, C., and Kim, B. S., "Adaptive Nonlinear Control System Design for Helicopter Robust Command Augmentation," *Aerospace Science and Technology*, Vol. 9, No. 3, 2005, pp. 241–251.
- [9] Dierks, T., and Jagannathan, S., "Output Feedback Control of a Quadrotor UAV Using Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 21, No. 1, 2010, pp. 50–66.
- [10] Isidori, A., *Nonlinear Control Systems*, Springer-Verlag, Berlin, 1995, pp. 247, 344.
- [11] Downs, J., Prentice, R., Dalzell, S., Besachio, A., Ivler, C. M., Tischler, M. B., and Mansur, M. H., "Control System Development and Flight Test Experience with the MQ-8B Fire Scout Vertical Take-off Unmanned Aerial Vehicle (VTUAV)," American Helicopter Society, May 2007.
- [12] Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E. "Inverted Autonomous Helicopter Flight via Reinforcement Learning," *Experimental Robotics IX*, edited by M. Ang and O. Khatib, Springer-Verlag, New York, 2006, pp. 363–372.
- [13] Gavrilets, V., Mettler, B., and Feron, E., "Human-Inspired Control Logic for Automated Maneuvering of a Miniature Helicopter," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 5, 2004, pp. 752–759.
- [14] Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y., "An Application of Reinforcement Learning to Aerobatic Helicopter Flight," *Neural Information Processing Systems Conference*, Dec. 2007.
- [15] Dierks, T., and Jagannathan, S., "Optimal Control of Affine Nonlinear Discrete-Time Systems," *Proceedings of the Mediterranean Conference on Control and Automation*, IEEE, Piscataway, NJ, 2009, pp. 1390–1395.
- [16] Dierks, T., and Jagannathan, S., "Optimal Control of Affine Nonlinear Continuous-Time Systems," *Proceedings of American Control Conference*, IEEE, Piscataway, NJ, 2010, pp. 1568–1573.
- [17] Mahoney, R., and Hamel, T., "Robust Trajectory Tracking for a Scale Model Autonomous Helicopter," *International Journal of Robust and Nonlinear Control*, Vol. 14, No. 12, 2004, pp. 1035–1059.
- [18] Lewis, F. L., and Syrmos, V. L., *Optimal Control*, 2nd ed., Wiley, Hoboken, NJ, 1995, pp. 333–339.
- [19] Lewis, F. L., Jagannathan, S., and Yesilderek, A., *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis, London, 1999, pp. 376–383.

Intelligent Constrained Optimal Control of Aerospace Vehicles with Model Uncertainties

Jie Ding* and S. N. Balakrishnan[†]

Missouri University of Science and Technology, Rolla, Missouri 65401

NOMENCLATURE

B	constant $n \times m$ matrix
d	unmodeled dynamics
\hat{d}	neural network approximation of the unmodeled dynamics
E_a	difference between the states of the actual plant and the states of the virtual plant
I	matrix of moment of inertias, kg-m^2
J	cost
J_k^a	actual cost (output of the critic network) of going from the time step k to the final stage
J_k^t	target cost of going from the time step k to the final stage
K	design parameter
k	time index
n	number of NNs for the unmodeled dynamics
p	total spacecraft angular momentum expressed in spacecraft coordinates
p^I	(constant) inertial angular momentum, N-m-s
Q, Q_W	weighting matrices for state
R, R_W	weighting matrices for control
r	commanded reference
t	time
U	control vector
U_d	desired steady-state control
U_{\max}	control constraint
W_i	weights of the i th uncertainty network
\hat{W}	critic network weights
X	state vector
X_a	states vector of the virtual plant

*Ph.D. Student, Department of Mechanical and Aerospace Engineering; jdr5c@mail.mst.edu.

[†]Professor, Department of Mechanical and Aerospace Engineering; bala@mst.edu.

X_d	desired state
X_N	final desired point
X_0	initial point
ε	accuracy
λ	costate vector
$\mu(\cdot)$	bounded, integrable, one-to-one, globally Lipschitz continuous function
τ	torque applied to the spacecraft by the reaction wheels motors, N-m
Φ	basis function of the critic network
ϕ, θ, ψ	three Euler angles describing the attitude of a spacecraft, rad
φ_i	basis function of the i th uncertainty network
Ψ	utility function
ω	angular velocity, rad/s

I. INTRODUCTION

Feedback control is the preferred solution for many systems because of its beneficial properties like robustness with respect to noise and modeling uncertainties. It is well-known that a dynamic programming formulation offers the most comprehensive solution approach to nonlinear optimal control in a state feedback form [1, 2]. However, solving the associated Hamilton–Jacobi–Bellman (HJB) equation demands a large (rather infeasible) number of computations and storage space dedicated to this purpose. An innovative idea was proposed in [3] to get around this numerical complexity by using a reinforcement-based approximate dynamic programming (ADP) formulation. The solution to the ADP formulation is obtained through a dual NN approach called the adaptive critic (AC). In one version of the AC approach, called the heuristic dynamic programming (HDP), one network (called the action network) represents the mapping between the state and control variables while a second network (called the critic network) represents the mapping between the state and the cost function to be minimized. Adaptive critic formulations can be found in many papers; some researchers have used the ADP formulations to solve problems with finite state spaces in applications to behavioral and computer sciences, operations research, and robotics [4–7]. Adaptive critics can also be considered reinforcement learning designs [4, 5]. These formulations employ primarily cost-function-based adaptive critics that we consider in this chapter. There are also many papers in the literature that use system science principles and neural networks to formulate the problems with applications to real-time feedback control of dynamic systems. A compendium of such applications can be found in Si et al. [8]. In recent years, many researchers have paid more attention to ADP in order to obtain approximate solutions of the HJB equation [9–12].

The use of NNs in feedback control systems was first proposed by Werbos [13]. Since then, NN control has been studied by many researchers. Recently, NN has entered the mainstream of control theory as a natural extension of adaptive control to systems that are nonlinear with tunable parameters. Overviews of the initial work in NN control are provided by [14] and the *Handbook of Intelligence Control* [3], which highlighted a host of difficulties to be addressed for closed-loop control applications. Basic problems that had to be addressed for closed-loop NN control [3, 14] included weight initialization for feedback stability, determining the gradients needed for backpropagation tuning, determining what to backpropagate, obviating the need for preliminary offline tuning, modifying backprop so that it tunes the weights forward through time, and providing efficient computer code for implementation. Recent works by Jagannathan and colleagues [15–17] have been useful in practical applications. Zheng and Jagannathan [15] investigated the use of NNs in obtaining nearly optimal solutions to the nonlinear discrete-time control problems. Yang and Jagannathan [16] applied an adaptive-critic-based controller to an atomic force microscope-based force controller to push nanoparticles on the substrates. He and Jagannathan [17] presented a reinforcement learning scheme in discrete time for the NN controller, where the action NN learning is performed based on a critic-supplied performance measure. Al-Tamimi et al. [18] use the HDP and the dual heuristic programming (DHP) structures to solve problems formulated with game theoretic notions. Vrabie et al. [19] proposed a new policy iteration technique to solve the online continuous-time linear quadratic regulator (LQR) problem for a partially model-free system (internal dynamics unknown). An online adaptive critic algorithm has been presented in which the actor performs continuous-time control, whereas the critic's correction of the actor's behavior is discrete in time until best performance is obtained. Shih et al. [20] formulated a novel reinforcement-learning-based output-adaptive NN controller to track a desired trajectory for a class of complex nonlinear discrete-time systems in the presence of bounded and unknown disturbances. Model-based synthesis of adaptive critic based controllers has been presented by Balakrishnan and Biega [21], Prokhorov and Wunsch [22], and Venayagamoorthy et al. [23] for systems driven by ordinary differential equations. Ferrari and Stengel [24] have implemented a global adaptive critic controller for a business jet. Lendaris et al. [25] have successfully shown in simulations that the HDP method can prevent cars from skidding when driving over unexpected patches of ice. In fact, there are many variants of the AC designs [22]. Typically, the AC designs are formulated in a discrete framework. Hanselman et al. consider the use of continuous time adaptive critics in [26].

While there has been a multitude of papers involving ACs, there is virtually no paper in the published literature that deals with the computational load (or alleviating it) associated with the AC designs. If the ADP formulation should find its way to engineering implementations, computationally efficient AC designs that show convergence are a priority. Balakrishnan's group (Padhi et al. [27, 28], Yadav et al. [29]) proposed a single network adaptive critic (SNAC) earlier.

However, that structure was based on a critic-based reinforcement learning network that outputs the costates as in the DHP. The problem in dealing with the costates is that they have no physical meaning in an engineering problem like the physical states of a system. Therefore, it is very difficult to get an idea of the magnitude of costates; in a multivariable problem, different costates can have values that could vary by several orders of magnitudes, thereby making convergence of the critic network very difficult. In contrast, the use of cost in a critic network as in this chapter is perhaps more meaningful. As opposed to the costates that have the same dimension as states in a multivariable problem and, therefore, demand a more diverse network structure and its training, the cost function is a *scalar*, and, therefore, the critic network dimension is minimal. The contribution of this chapter is an implementable cost-based reinforcement learning structure for solving constrained optimal control problems. It captures the mapping between the states (at time k) and the optimal cost (from k to the end). Note that although costates have no physical meaning, the output of the cost network provides valuable information to the designer and an idea of the remaining cost involved at any stage of the process as a function of the states of the system. In fact, the cost-function-based single network adaptive critic (J-SNAC) proposed in this chapter is applicable to the type of control-affine nonlinear problems presented in some recent papers [30, 31] while saving about 50% computation time as compared to the typical HDP structure used in those papers due to the elimination of one network; interested readers can find simulation comparisons in [32].

Many real-life problems have controller limits. Control inputs may have maximum or minimum values. For example, valves in chemical process control [33] have a maximum displacement (when fully open) and a minimum displacement (when fully closed). Methods for solving constrained control problems are presented by Ryan [34] and Bernstein [35]. Bernstein [35] developed the optimal saturating feedback control laws involving bang-bang action, which is a modification of the control laws given by Frankena et al. [36] and Ryan [34]. In [37], a method of analyzing a system for stability with control magnitude constraints has been presented, in which stability is shown as a tradeoff between the sizes of the initial condition set and the size of the control constraints. Bernstein [38] presented the optimal saturating feedback control laws by using steady-state HJB theory, and the control law is shown to be optimal for a modified performance functional with a discontinuous integrand. Yang and Wu [39] proposed a kind of neural network for solving constrained optimal control problems. It is shown that under suitable conditions the state trajectory of the neural network converges to an equilibrium point, which corresponds to a local optimal solution to the original problem. Lyshevski [40, 41] outlined a constrained optimization framework to solve optimization problems for nonlinear time-varying systems with state and control bounds. It is not generally easy to create a parameterized functional equation for the optimal cost as he has done. In [42], the ability of the constrained optimization concept to design the bounded controller for multivariable advanced aircraft was demonstrated. In [43], an HJB equation-based

constrained optimal control algorithm is proposed for a bilinear system. In [44], fixed-final time constrained input optimal control laws using neural networks to solve HJB equations for general affine in the input nonlinear systems are proposed. Although many methods have been proposed to deal with the constrained optimal problem, solving the associated HJB equation demands a large number of computations and storage space. To deal with this problem, in this chapter a reinforcement training-based cost function synthesis using a single network adaptive critic (J-SNAC) technique is presented to solve optimal nonlinear constrained control problems with model uncertainties. A nonquadratic cost function [41] is used that incorporates the control constraints. Necessary equations for optimal control are derived, and an algorithm to solve the constrained-control problem with J-SNAC is developed.

The rest of the chapter is organized as follows. In Sec. II, the approximate dynamic programming equations are derived. In Sec. III, the J-SNAC technique with a nonquadratic cost function is developed. An online updated NN to calculate the uncertainty as a part of an approximated plant is discussed in Sec. IV. Formulation of optimal tracking problems with input constraints is presented in Sec. V. Numerical results from a spacecraft problem with torque constraints and an aircraft problem with actuator limits are given in Sec. VI. Conclusions are presented in Sec. VII.

II. APPROXIMATE DYNAMIC PROGRAMMING

In this section, the approximate (discrete) dynamic programming equations, which both the AC and the J-SNAC approaches seek to solve, are presented. An interested reader can find more details about the derivations in [3, 21]. Note that a prime requirement to apply the AC or the J-SNAC method is to formulate the problem in discrete time. If the original problems are described with continuous dynamics, the control designer has the freedom to use any appropriate discretization scheme. For example, one can use the Euler approximation for the state equation and trapezoidal approximation for the cost function [45]. In a discrete-time formulation, we want to find an admissible control U_k , which causes the system given by

$$X_{k+1} = F_k(X_k, U_k) \quad (1)$$

to follow an admissible trajectory from an initial point X_0 to a final desired point X_N while minimizing a desired cost function J given by

$$J = \sum_{k=0}^{N-1} \Psi_k(X_k, U_k) \quad (2)$$

where $X_k \in \mathbb{R}^n$ and $U_k \in \mathbb{R}^m$ are the state and control vectors at time step k . The functions F and Ψ_k are assumed to be differentiable with respect to both X_k and

U_k . Moreover, Ψ_k is assumed to be convex (e.g., a quadratic function in X_k and U_k). One can notice that when $N \rightarrow \infty$, this cost function represents a regulator (infinite time) problem. The steps in obtaining optimal control at every step are now described.

Remark 1: Control U_k must both stabilize the system on a compact domain $\Omega \subset \mathbb{R}^{n \times 1}$ and make the cost functional value finite so that the control is admissible [46].

The cost function in Eq. (2) is rewritten to start from time step k as

$$J_k = \sum_{\bar{k}=k}^{N-1} \Psi_{\bar{k}}(X_{\bar{k}}, U_{\bar{k}}) \quad (3)$$

The cost J_k can be split into two terms as

$$J_k = \Psi_k + J_{k+1} \quad (4)$$

where Ψ_k and $J_{k+1} = \sum_{\bar{k}=k+1}^{N-1} \Psi_{\bar{k}}$ represent the “utility function” at time step k and the cost-to-go from time step $k+1$ to N , respectively.

The $n \times 1$ costate vector (Lagrange’s multiplier) at time step k is defined as

$$\lambda_k = \partial J_k / \partial X_k \quad (5)$$

The necessary condition for optimality is obtained by setting the first derivative of J_k with respect to U_k to zero as

$$\partial J_k / \partial U_k = 0 \quad (6)$$

Equation (6) can be further expanded as

$$\frac{\partial J_k}{\partial U_k} = \frac{\partial \Psi_k}{\partial U_k} + \frac{\partial J_{k+1}}{\partial U_k} = \frac{\partial \Psi_k}{\partial U_k} + \left(\frac{\partial X_{k+1}}{\partial U_k} \right)^T \frac{\partial J_{k+1}}{\partial X_{k+1}} \quad (7)$$

The optimal control equation can, therefore, be written as

$$\frac{\partial \Psi_k}{\partial U_k} + \left(\frac{\partial X_{k+1}}{\partial U_k} \right)^T \frac{\partial J_{k+1}}{\partial X_{k+1}} = 0 \quad (8)$$

The costate equation is derived by equating the partial derivatives of both sides in Eq. (4) with respect to X_k as

$$\lambda_k = \frac{\partial \Psi_k}{\partial X_k} + \frac{\partial J_{k+1}}{\partial X_k} = \frac{\partial \Psi_k}{\partial X_k} + \left(\frac{\partial X_{k+1}}{\partial X_k} \right)^T \frac{\partial J_{k+1}}{\partial X_{k+1}} = \frac{\partial \Psi_k}{\partial X_k} + \left(\frac{\partial X_{k+1}}{\partial X_k} \right)^T \lambda_{k+1} \quad (9)$$

Equations (1), (8), and (9) have to be solved simultaneously, along with appropriate boundary conditions to obtain optimal control. For the regulator problems, the boundary conditions usually take the form: X_0 is fixed and $\lambda_N \rightarrow 0$ as $N \rightarrow \infty$. Note that J-SNAC is applicable to the class of problems where the state equation is control-affine with a cost function that yields an explicit solution for the control variable in terms of the state and the cost variables from Eq. (8).

III. J-SNAC SYNTHESIS WITH NONQUADRATIC COST

In this section, the J-SNAC technique is discussed in detail. Approximate dynamic programming by its formulation and development is discrete. Therefore, the developments in this paper are all in a discrete-time formulation. Moreover, a discrete-time development will help in possible implementation scenarios where the measurements and control computations occur at certain frequencies and not in a continuous mode. In the J-SNAC design, a critic network captures the functional relationship between state X_k and cost J_k .

A. HJB EQUATION WITH CONSTRAINTS ON THE CONTROL INPUT

In a discrete-time formulation, the system dynamics are described by

$$X_{k+1} = f(X_k) + BU_k \quad (10)$$

where $X_k \in \mathbb{R}^n$ and $U_k \in \mathbb{R}^m$ are the state and control vectors, $f(\cdot) \in \mathbb{R}^n$ is a smooth mapping, and $B \in \mathbb{R}^{n \times m}$.

The following cost function is used in the design of constrained controllers [47]:

$$J = \sum_{k=0}^{N \rightarrow \infty} \Psi_k \quad (11)$$

where the utility function $\Psi_k = \left\{ \frac{1}{2} X_k^T Q_W X_k + \int_0^{U_k} [\mu^{-1}(v)]^T R_W dv \right\} \Delta t$, and $Q_W \in \mathbb{R}^{n \times n}$ and $R_W \in \mathbb{R}^{m \times m}$ are the diagonal weighting matrices, and $\mu(\cdot)$ is a bounded, integrable, one-to-one, real-analytic globally Lipschitz continuous function $\mu \in \mathbb{R}^m$. A function f is called Lipschitz continuous if there exist a real constant $L \geq 0$ such that, for all x and y in \mathbb{R} , $|f(y) - f(x)|/|y - x| \leq L$ [48]. The reason for using this form of controller cost will become clear when the expression for control is obtained. Note that Δt is the sampling interval.

The optimality condition in Eq. (8) leads to

$$\mu^{-1}(U_k) \Delta t R_W + \left(\frac{\partial X_{k+1}}{\partial U_k} \right)^T \frac{\partial J_{k+1}}{\partial X_{k+1}} = 0 \quad (12)$$

From Eq. (12), the expression for the constrained optimal control is obtained as

$$\begin{aligned} U_k &= -\mu \left[(\Delta t R_W)^{-1} B^T \frac{\partial J_{k+1}}{\partial X_{k+1}} \right] \\ &= -\mu \left\{ (\Delta t R_W)^{-1} B^T \left[\left(\frac{\partial X_{k+1}}{\partial X_k} \right)^T \right]^{-1} \left(\frac{\partial J_k}{\partial X_k} - \Delta t Q_W X_k \right) \right\} \end{aligned} \quad (13)$$

where it is assumed that $\left[\left(\frac{\partial X_{k+1}}{\partial X_k} \right)^T \right]^{-1}$ exists. All a control designer needs to do to impose controller limits is to choose $\mu(\cdot)$ properly.

By using Eq. (9), the costate equation can be obtained as

$$\lambda_k = \Delta t Q_W X_k + [\partial X_{k+1} / \partial X_k]^T \lambda_{k+1} \quad (14)$$

B. NEURAL NETWORK TRAINING

In the J-SNAC synthesis with nonquadratic cost approach, the steps for the reinforcement-driven training of the critic network, which captures the relationship between X_k and J_k , are as follows (Fig. 1):

1. Input X_k to the critic network to obtain $J_k = J_k^a$.
2. Calculate $\lambda_k = \partial J_k / \partial X_k$, and λ_{k+1} by Eq. (14).
3. Calculate U_k , and form the optimal control equation (13).
4. Use X_k and U_k to get X_{k+1} from Eq. (1).
5. Input X_{k+1} to the critic network to get J_{k+1} .

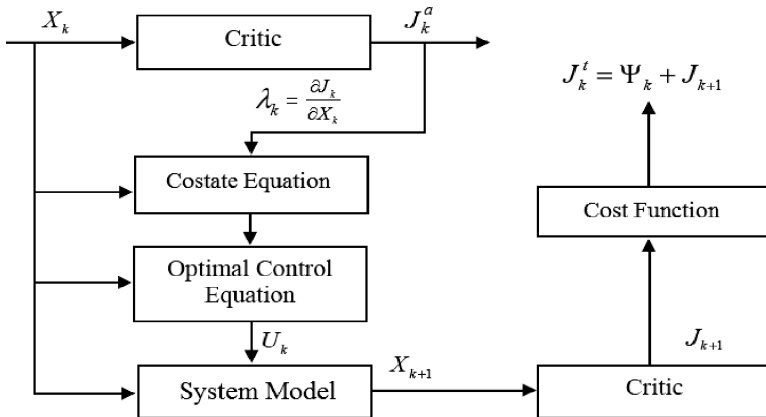


FIG. 1 J-SNAC network training scheme.

6. Use X_k , U_k , and J_{k+1} , to calculate J_k^t with Eq. (4).
7. Train the critic network by solving Eq. (A.10) for network weights (see Appendix).
8. Check the convergence of the critic network [by defining the relative error $e_c \equiv (\|J_k^t - J_k^a\| / \|J_k^t\|)$]; the training process is stopped when $\|e_c\| < tol_c$; otherwise, repeat steps 1–8.

A numerical method used for the J-SNAC training is presented in the Appendix.

IV. DYNAMIC REOPTIMIZATION OF J-SNAC CONTROLLER

In this section, we consider the plant dynamics with parametric uncertainties or unmodeled nonlinearities. This situation might happen when a low-fidelity, mathematically tractable model is used for controller design or when unmodeled uncertainties are present. In this case, the original controller is no longer optimal for the true dynamics. We discuss the dynamic reoptimization of the J-SNAC controller for the changed dynamics as the unmodeled part of the dynamics is learned online. This is achieved with the help of a “virtual plant” that is similar to the actual plant. The uncertainty approximation is achieved by using an online neural network. This is because NNs 1) are excellent function approximators and 2) lend themselves to analytical treatment, a fact that helps with proofs on error bounds, stability, etc.

Let the actual plant have the structure [different from Eq. (10)]

$$X_{k+1} = f(X_k) + BU_k + d(X_k) \quad (15)$$

where the controller U_k will have to be reoptimized with the unmodeled dynamics $d(X_k)$ present. Because the term $d(X_k)$ in the plant equation is unknown, the first step in the controller reoptimization is to approximate the uncertainty in the plant equation. For this purpose a virtual plant is defined. Let X_a represent the vector of states of the virtual plant.

The dynamics of this virtual plant is governed by

$$X_{a_{k+1}} = f(X_k) + BU_k + \hat{d}(X_k) + K(X_k - X_{a_k}), X_a(0) = X(0) \quad (16)$$

where $K > 0$ is an $n \times n$ stable design parameter matrix. It is assumed that all of the actual plant states X are available for measurement at every step. The term $\hat{d}(X_k)$ is the neural network approximation of the actual plant. Subtracting Eq. (16) from Eq. (15), by defining $E_{a_k} \equiv X_{a_k} - X_k$, we obtain $X_{a_{k+1}} - X_{k+1} = \hat{d}(X_k) - d(X_k) + K(X_k - X_{a_k})$ or $E_{a_{k+1}} = \hat{d}(X_k) - d(X_k) - KE_{a_k}$. It can be seen that as $\hat{d}(X_k) - d(X_k)$ approaches zero, the expression of E_a is exponentially stable, that is, $E_a \rightarrow 0$ as $t \rightarrow \infty$. The J-SNAC dynamic reoptimization scheme is shown in Fig. 2.

Defining $X = [x_1, x_2, \dots, x_n]^T$, $E_a = [e_{a1}, e_{a2}, \dots, e_{an}]^T$, and $d(X) = [d_1(X), d_2(X), \dots, d_n(X)]^T$, where $d_i(X)$ denotes the unmodeled dynamics in

the differential equation for the i th state of the system. This study assumes “ n ” NNs (one for each component of the unmodeled dynamics) so as to allow for a simpler development and analysis. This is presented in Fig. 3. Separating each channel, the state equations become

$$x_{i_{k+1}} = f_i(X_k) + b_i U_k + d_i(X_k) \quad (17)$$

$$x_{ai_{k+1}} = f_i(X_k) + b_i U_k + \hat{d}_i(X_k) + K_i e_{ai_k} \quad (18)$$

where $e_{ai_k} \equiv x_{ai_k} - x_{i_k}$, K_i is the i th row of the matrix K . Subtracting Eq. (18) from Eq. (17), we obtain

$$x_{ai_{k+1}} - x_{i_{k+1}} = \hat{d}_i(X_k) - d_i(X_k) - K_i e_{ai_k} \quad (19)$$

How do we approximate the uncertainty $d_i(X_k)$ with a neural network? Different architectures for neural networks exist in the literature [49–51]. In this chapter, a linear-in-the-parameter network is chosen. The reasons are twofold: keep the architecture simple and *mathematically tractable*. Let there exist a neural network with an optimum set of weights that approximates $d_i(X)$ within a certain accuracy of ε_i in a compact domain for the states. That is,

$$d_i(X_k) = W_{i_k}^T \varphi_i(X_k) + \varepsilon_i \quad (20)$$

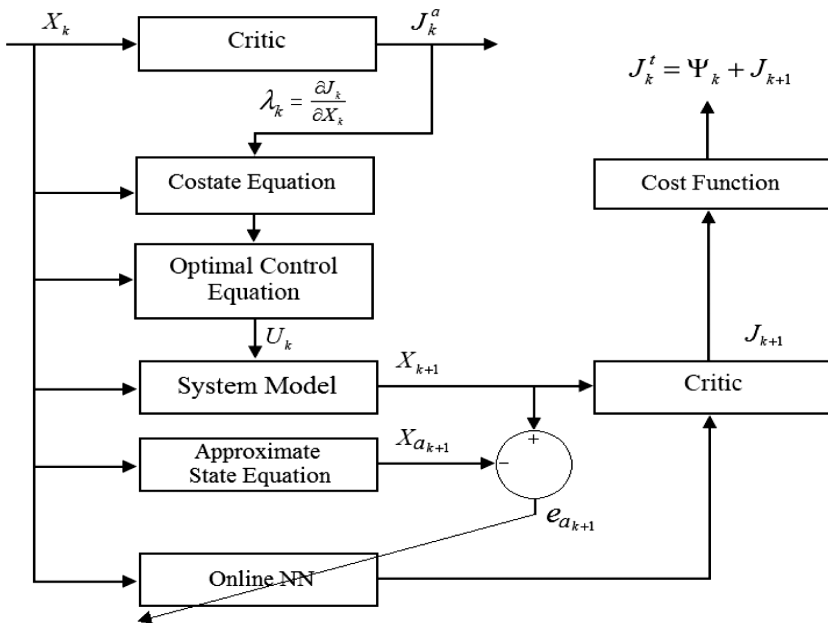


FIG. 2 J-SNAC dynamic reoptimization scheme.

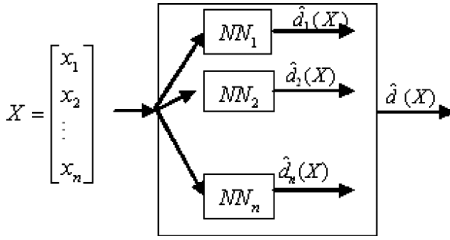


FIG. 3 Uncertainty neural network structure.

In Eq. (20), $\varphi_i()$ represents a set of basis functions used in the neural network approximations. Choosing proper basis functions for a given problem is still an art and not a science. In applications, one examines the system model

and selects the number and form of the basis functions from the states of the system. Note that $\hat{d}_i(X_k) = \hat{W}_{ik}^T \varphi_i(X_k)$, where $\hat{W}_{ik}^T \varphi_i(X_k)$ is the output of the uncertainty approximation neural network. \hat{W}_{ik} represents the uncertainty approximation neural network weights.

Substituting Eq. (20) into Eq. (19), we obtain

$$x_{ai_{k+1}} - x_{i_{k+1}} = \hat{W}_{ik}^T \varphi_i(X_k) + \varepsilon_i - W_{ik}^T \varphi_i(X_k) - e_{ai_k} \quad (21)$$

$$e_{ai_{k+1}} = x_{ai_{k+1}} - x_{i_{k+1}} = \tilde{W}_{ik}^T \varphi_i(X_k) + \varepsilon_i - e_{ai_k} \quad (22)$$

where $\tilde{W} = W_i - \hat{W}_i$ is the difference between the optimal weights of the neural network that represents $d_i(X_k)$ and the actual network weights. More details on the update rule can be found in [52–54]. Furthermore, we assume that the basis functions are locally Lipschitz in a compact domain that contains the state-space limits of the problem.

V. TRACKING PROBLEM WITH INPUT CONSTRAINT

In Sec. III, equations were developed that are directly applicable to nonlinear optimal regulator control problems with control constraints. This section extends J-SNAC applicability to constrained *tracking* problems. To accomplish that, the cost function is rewritten in terms of the errors in the states X_k and control U_k from the desired values of states X_{d_k} and control required U_{d_k} to keep the states at the desired values. The system dynamics used is the same as before in Eq. (10). To deal with the tracking problem with controller limits, a non-quadratic cost function is assumed as

$$J = \sum_{k=0}^{N-\infty} \left\{ \frac{1}{2} (X_k - X_{d_k})^T Q_W (X_k - X_{d_k}) + \int_0^{U_k - U_{d_k}} [\mu^{-1}(U_k - U_{d_k})]^T R_W d(U_k - U_{d_k}) \right\} \Delta t \quad (23)$$

The control objective is to drive the square of the error between X_k and X_{d_k} and square of the error between U_k and U_{d_k} to zero. This objective makes the formulation of J finite, and, in fact, the tracking problem is just recast as a regulator problem.

In this case, the optimality condition leads to the control equation

$$\begin{aligned} U_k &= U_{d_k} + \mu \left[-(\Delta t R_W)^{-1} B^T \frac{\partial J_{k+1}}{\partial X_{k+1}} \right] \\ &= U_{d_k} + \mu \left\{ -(\Delta t R_W)^{-1} B^T \left[\left(\frac{\partial X_{k+1}}{\partial X_k} \right)^T \right]^{-1} \left(\frac{\partial J_k}{\partial X_k} - \Delta t Q_W X_k \right) \right\} \quad (24) \end{aligned}$$

For the applications considered in this study, the operator μ is selected as (for each control component)

$$\mu(\cdot) = \frac{a - b e^{-2(\cdot)}}{c + l e^{-2(\cdot)}} \quad (25)$$

and by selecting $a = b = U_{\max} + U_{d_k}$, $c = (U_{\max} + U_{d_k}) / (U_{\max} - U_{d_k})$, $l = 1$, the function μ has the upper limit $(U_{\max} - U_{d_k})$ and the lower limit $(-U_{\max} - U_{d_k})$ so that the control U_k in Eq. (24) has the upper limit U_{\max} and the lower limit $-U_{\max}$. It is assumed that the desired control U_{d_k} does not exceed the constraint U_{\max} (for $U_k \in \mathbb{R}^m$, ϕ_i ($i = 1, 2, \dots, m$) is selected as $\mu_i(\cdot) = (a - b e^{-2(\cdot)}) / (c + l e^{-2(\cdot)})$, where $a = b = U_{i_{\max}} + U_{i_{d_k}}$, $c = (U_{i_{\max}} + U_{i_{d_k}}) / (U_{i_{\max}} - U_{i_{d_k}})$, $l = 1$).

The costate equation for the tracking problem is obtained as

$$\lambda_k = \Delta t Q_W (X_k - X_{d_k}) + [\partial X_{k+1} / \partial X_k]^T \lambda_{k+1} \quad (26)$$

VI. NUMERICAL RESULTS

In this section, the developed J-SNAC approach is used to synthesize optimal constrained controllers in two aerospace example problems. The first application is a spacecraft control problem, and the second is an aircraft control problem.

A. EXAMPLE: SPACECRAFT CONTROL

1. PROBLEM DESCRIPTION AND OPTIMALITY CONDITIONS

Consider a general rigid spacecraft controlled by reaction wheels. It is assumed that the control torques are applied through a set of three reaction wheels along the spacecraft's three orthogonal axes. In an inertial frame, the dynamic rotational

motion equation [55] of the spacecraft is given by

$$I\dot{\omega} = p \times \omega + \tau \quad (27)$$

Choosing the rotational states $x = [\phi, \theta, \psi]^T$, where ϕ, θ, ψ are the three Euler angles describing the attitude of a spacecraft, the kinematic equations of the spacecraft are written as

$$\dot{x} = J(x)\omega \quad (28)$$

where

$$J(x) = \begin{bmatrix} 1 & \sin(\phi) \tan(\theta) & \cos(\phi) \tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) \sec(\theta) & \cos(\phi) \sec(\theta) \end{bmatrix} \quad (29)$$

The total spacecraft angular momentum p is written as

$$p = R(x)p^I \quad (30)$$

where $p^I = [1, -1, 0]^T$ is the (constant) inertial angular momentum and

$$R(x) = \begin{bmatrix} \cos(\psi) \cos(\theta) & \cos(\psi) \sin(\theta) \sin(\phi) - \sin(\psi) \cos(\phi) & \cos(\psi) \sin(\theta) \sin(\phi) + \sin(\psi) \sin(\phi) \\ \sin(\psi) \cos(\theta) & \sin(\psi) \sin(\theta) \sin(\phi) + \cos(\psi) \cos(\phi) & \sin(\psi) \sin(\theta) \cos(\phi) - \cos(\psi) \sin(\phi) \\ -\sin(\psi) & \cos(\psi) \sin(\theta) & \cos(\psi) \cos(\theta) \end{bmatrix} \quad (31)$$

Choosing $X = [\phi, \theta, \psi, \omega_x, \omega_y, \omega_z]^T$ as the state vector and $U = [\tau_1, \tau_2, \tau_3]^T$ as the control vector, and assuming an uncertainty $d = [d_1, d_2, 0]^T$ to be present in the system, the dynamics of the system are characterized by

$$\begin{cases} \dot{x} = J(x)\omega \\ \dot{\omega} = I^{-1}p \times \omega + I^{-1}\tau + d \end{cases} \quad (32)$$

More explicitly, Eq. (32) can be rewritten as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} 0_{3 \times 3} & J(x) \\ 0_{3 \times 3} & I^{-1}p \times \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \psi \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \begin{bmatrix} 0_{3 \times 3} \\ I^{-1} \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ d_1 \\ d_2 \\ 0 \end{bmatrix} \quad (33)$$

where $d_1 = 0.1\omega_x^3$, $d_2 = 0.5 \sin(\omega_x) \cos^2(\omega_y)$ are the assumed form of uncertainties. The control objective is to drive all of the states to zero as $t \rightarrow \infty$.

A nonquadratic cost function J_c with an input constraint is selected as

$$J_c = \int_0^{\infty} \left\{ \frac{1}{2} X^T Q_W X + \int [\mu^{-1}(U_{\max}^{-1} U)]^T R_W dU \right\} dt \quad (34)$$

where $Q_W \geq 0$ and $R_W > 0$ are weighting matrices for the state and control, respectively, and U_{\max} is the control constraint. For use with the J-SNAC, the spacecraft dynamics and the cost function need to be described in a discrete form. Because the generic equations in Sec. II have to be changed to include the problem-oriented construction, all of the relevant equations are presented.

The state equation is discretized as

$$X_{k+1} = X_k + \Delta t [f(X_k) + B U_k] \quad (35)$$

The nonquadratic cost function (34) is discretized as

$$J_d = \sum_{k=0}^{N \rightarrow \infty} \left\{ \frac{1}{2} X_k^T Q_W X_k + \int_0^{U_k} [\mu^{-1}(U_{\max}^{-1} U)]^T R_W dU \right\} \Delta t \quad (36)$$

The optimality condition leads to the control equation

$$\begin{aligned} U_k &= U_{\max} \mu \left[-(\Delta t R_W)^{-1} B^T \frac{\partial J_{k+1}}{\partial X_{k+1}} \right] \\ &= U_{\max} \mu \left\{ -(\Delta t R_W)^{-1} B^T \left[\left(\frac{\partial X_{k+1}}{\partial X_k} \right)^T \right]^{-1} \left(\frac{\partial J_k}{\partial X_k} - \Delta t Q_W X_k \right) \right\} \end{aligned} \quad (37)$$

The costate equation can be obtained as

$$\lambda_k = \Delta t Q_W X_k + \left[\frac{\partial F_k}{\partial X_k} \right]^T \lambda_{k+1} \quad (38)$$

where F_k represents the expression on the right-hand side of Eq. (35).

In this problem, the sample time $\Delta t = 0.005$ s; Q_W is selected as a diagonal matrix with $\text{diag}(20, 20, 20, 0, 0, 0)$. Note that the first three elements are the variables of interest that should be driven to zero. The values for the controller weighting matrix R_W are selected as $\text{diag}(10^{-3}, 10^{-3}, 10^{-3})$. The Lipschitz continuous function μ is selected as $\mu(\cdot) = \tanh(\cdot)$.

In the J-SNAC synthesis, the cost J_k is a function of the network weights \hat{W}^k and states X_k , which is given by Eq. (A.2) as $J_k = \hat{W}^{kT} \Phi(X_k)$. In this problem, the network weights were initialized to zero. The basis functions $\Phi(X)$ are selected as

$$\begin{bmatrix} X_1, X_2, \dots, X_6, X_1^2, X_2^2, \dots, X_6^2, X_1 X_2, X_1 X_3, X_1 X_4, X_1 X_5, X_1 X_6, \\ X_2 X_3, X_2 X_4, X_2 X_5, X_2 X_6, X_3 X_4, X_3 X_5, X_3 X_6, X_4 X_5, X_4 X_6, X_5 X_6 \end{bmatrix}$$

and m , the number of sets in Eq. (A10) for the network synthesis is selected as 100.

2. UNCERTAINTY ESTIMATION

In this study, the uncertainty neural network structure is $\hat{d}_i = W_i^T \varphi$, $i = 1, 2$. The design parameter K is selected as 10. W_1 and W_2 are both initialized as 27×1 random vectors. The basis functions are selected as $\varphi = \text{kron}[\text{kron}(c_1, c_2), c_3]$, where $c_1 = [1 \sin(\omega_x) \cos(\omega_x)]^T$, $c_2 = [1 \sin(\omega_y) \cos(\omega_y)]^T$, $c_3 = [1 \sin(\omega_z) \cos(\omega_z)]^T$ and kron denotes Kronecker product.

The discretized equations can be written as

$$X_{k+1} = X_k + \Delta t[f(X_k) + BU_k + d_k] \quad (39)$$

Expression for optimal control is the same as Eq. (37). Because of the presence of uncertainties in Eq. (39), the costate equation changes to

$$\lambda_k = \Delta t Q_W X_k + [\partial F'_k / \partial X_k]^T \lambda_{k+1} \quad (40)$$

where F'_k represents the expression on the right-hand side of Eq. (39), which also accounts for the uncertainty. During each iteration of the simulation, the critic network is updated. Because d_k is unknown \hat{d}_k , the output of the online neural

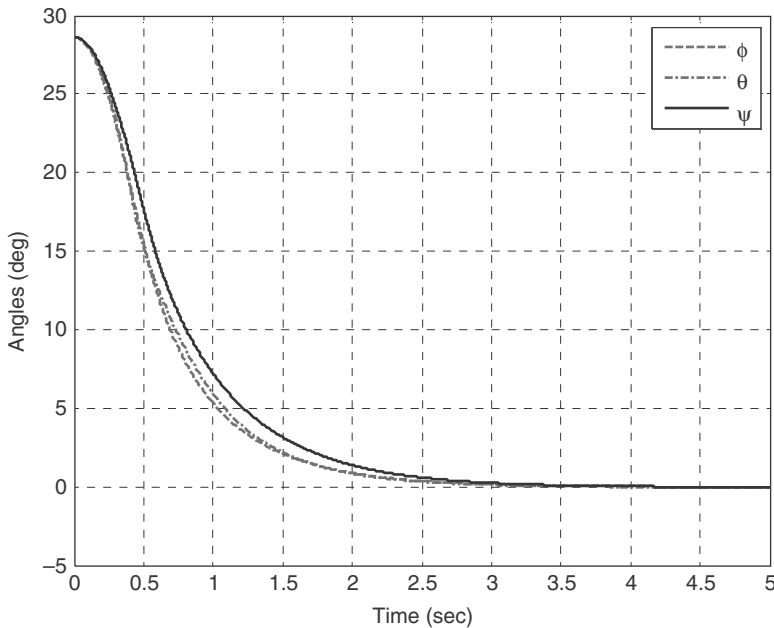


FIG. 4 Histories of angles describing the attitude.

network is used. The online training is carried out at every instant k by using X_k as the input to the cost network and obtaining the new target cost J_k^t as the output.

3. ANALYSIS OF RESULTS

Time histories of the Euler angles and the spacecraft rotation rates are presented in Figs. 4 and 5, respectively. It can be seen that all of the states go to zero within 5 s.

In this study, the following steps are used to simulate controller limits and show the J-SNAC performance. First, the program is run without the control constraint to find the unconstrained maximum control value U_m . This was found to be $U_m = [\tau_{1m}, \tau_{2m}, \tau_{3m}]^T = [71, 71, 69]^T$ (N·m). To demonstrate the efficacy of the proposed constrained optimal control technique, the control constraint U_{\max} is selected as 50% of U_m . The constrained control plots are given in Fig. 6. It can be seen that at the very beginning the torque values stay at the limit $U_{\max} = 0.5U_m = [35.5, 35.5, 34.5]^T$ (N·m), and are within the limits throughout afterward.

True and estimated uncertainty histories are shown in Fig. 7. It can be seen that the estimated uncertainties nicely and quickly track the true uncertainties. Because the uncertainties have been assumed to be functions of the spacecraft's

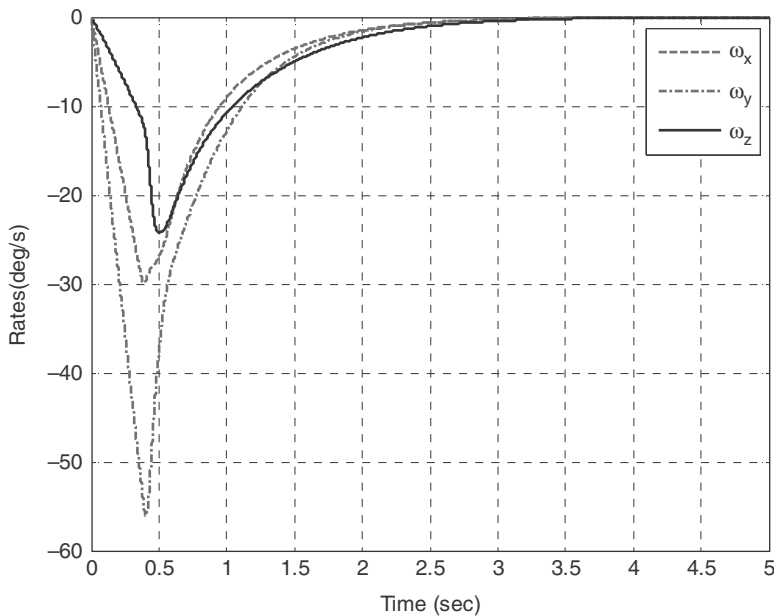


FIG. 5 Histories of spacecraft angular rates.

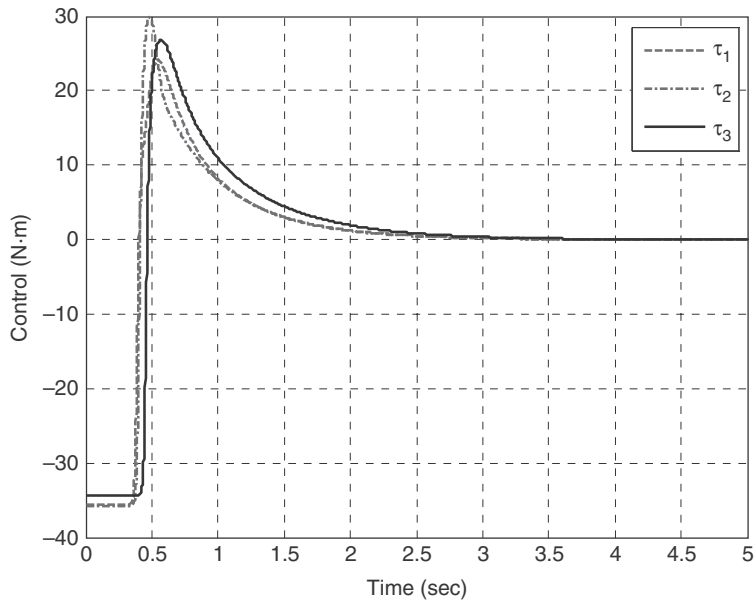


FIG. 6 Constrained control histories.

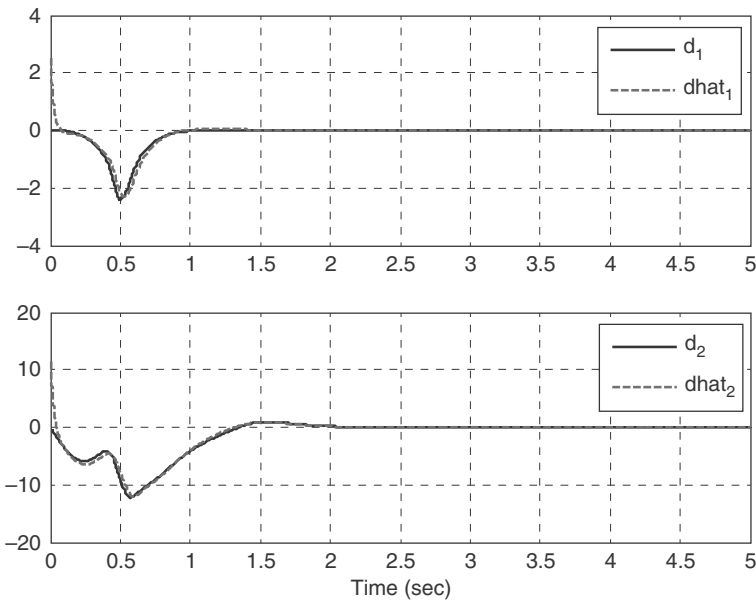


FIG. 7 True and estimated uncertainties' histories.

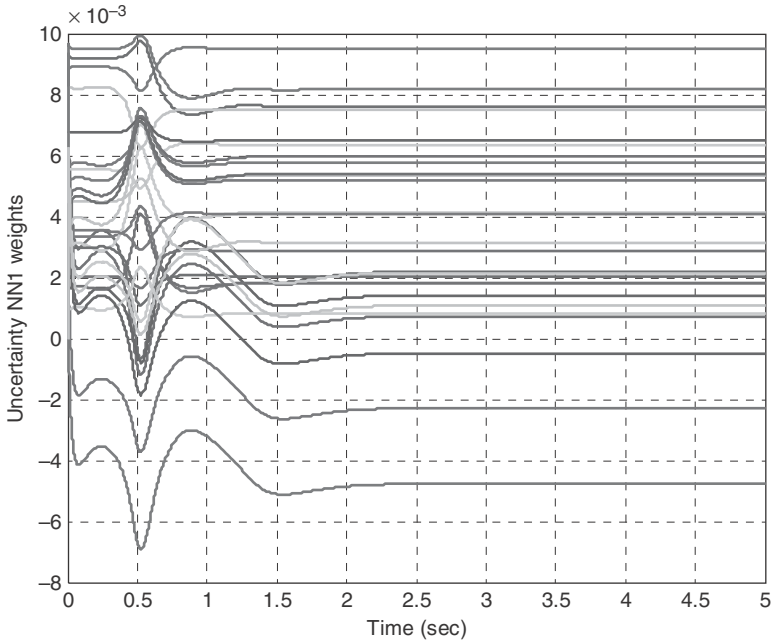


FIG. 8 Weights history.

angular velocities, they go to zero as expected. Histories of the weights of the two online neural networks are given in Figs. 8 and 9.

B. EXAMPLE: F-16 SHORT-PERIOD DYNAMICS FLIGHT CONTROL

1. PROBLEM DESCRIPTION AND OPTIMALITY CONDITIONS

Performance of the J-SNAC in a tracking scenario is examined in this section. A longitudinal dynamics model [56] is used in this chapter. It is given by

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -1.019 & 1 \\ 0.8223 & -1.0774 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} 0 \\ -0.1756 \end{bmatrix} [\delta_e + f(\alpha, \delta_e)] + d(\alpha) \quad (41)$$

In this problem, the control objective is to adjust the elevator deflection δ_e so that the angle of attack α tracks a commanded reference input $r(t)$, which is given as

$$r(t) = 0.2 \left(\frac{1}{1 + e^{t-8}} - \frac{0.5}{1 + e^{t-8}} + \frac{1}{1 + e^{t-20}} - e^{-0.2t} - 0.5 \right) \quad (42)$$

The uncertainties included in this problem are given by

$$d(\alpha) = [-1.28\alpha^3 0]^T$$

$$f(\alpha, \delta_e) = \left[(1 - C_0)e^{\frac{(\alpha - \alpha_0)^2}{2\sigma^2}} + C_0 \right] [\tanh(\delta_e + h) + \tanh(\delta_e - h) + 0.01]\delta_e$$

where $\sigma = 0.25$, $C_0 = 0.1$, $h = 0.14$, and $\alpha_0 = 0$. Note that $d(\alpha)$ is an *unmatched uncertainty* and $f(\alpha, \delta_e)$ is a *matched uncertainty*. Both needed to be estimated for computing the new optimal control.

Defining $X = [x_1 \ x_2]^T = [\alpha \ q]^T$, $U = \delta_e$, $d = [d_1 \ d_2]^T = Bf(\alpha, \delta_e) + d(\alpha)$ with $B = \begin{bmatrix} 0 \\ -0.1756 \end{bmatrix}$, the system dynamics is given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -1.019 & 1 \\ 0.8223 & -1.0774 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -0.1756 \end{bmatrix} U + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \quad (43)$$

The state equation in (43) is discretized in form similar to Eq. (35). A nonquadratic cost function with an input constraint is selected as in

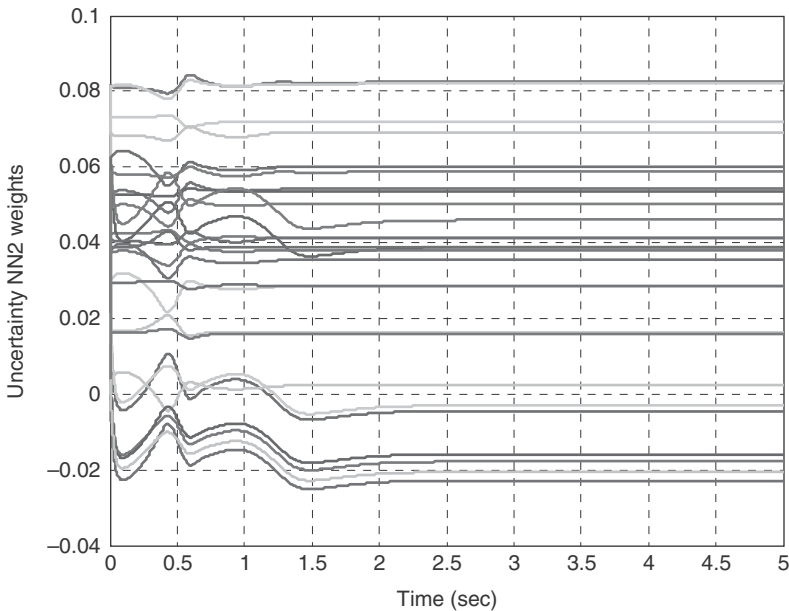


FIG. 9 Weights history.

Eq. (23), in which the function μ is selected as $\mu(U_k) = \frac{a - be^{-2U_k}}{c + le^{-2U_k}}$ with $a = b = U_{\max} + U_{d_k}$, $c = \frac{U_{\max} + U_{d_k}}{U_{\max} - U_{d_k}}$, $l = 1$.

The optimality condition leads to the control equation, which is the same as Eq. (24) and is repeated here for convenience:

$$\begin{aligned} U_k &= U_{d_k} + \mu \left[-(\Delta t R_W)^{-1} B^T \frac{\partial J_{k+1}}{\partial X_{k+1}} \right] \\ &= U_{d_k} + \mu \left\{ -(\Delta t R_W)^{-1} B^T \left[\left(\frac{\partial X_{k+1}}{\partial X_k} \right)^T \right]^{-1} \left(\frac{\partial J_k}{\partial X_k} - \Delta t Q_W X_k \right) \right\} \end{aligned} \quad (24)$$

The costate equation can be obtained as

$$\lambda_k = \Delta t Q_W (X_k - X_{d_k}) + \left[\frac{\partial F_k}{\partial X_k} \right]^T \lambda_{k+1} \quad (44)$$

where F_k represents the expression on the right-hand side of Eq. (35), $\Delta t = 0.005$ s, $Q_W = \text{diag}(1, 1)$, and $R_W = 0.005$. The input constraint U_{\max} is selected as 12 deg.

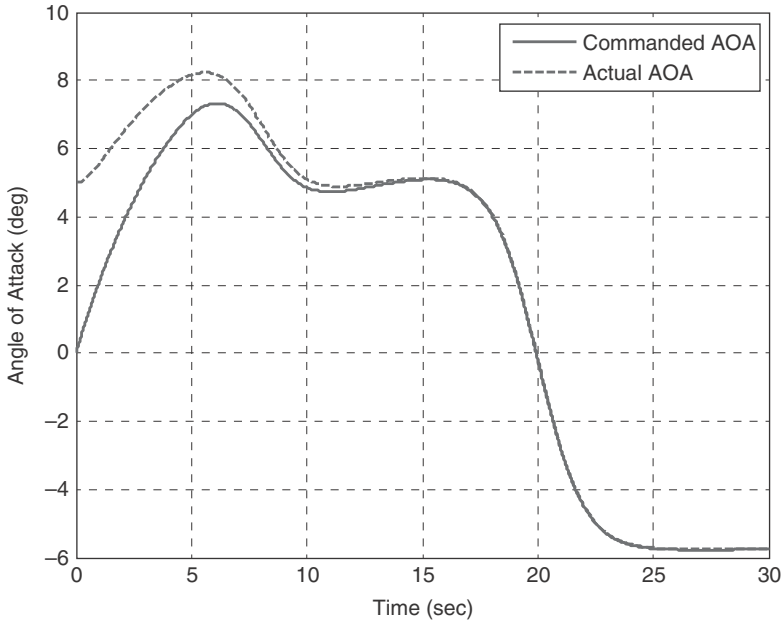


FIG. 10 Histories of commanded and actual angle of attack.

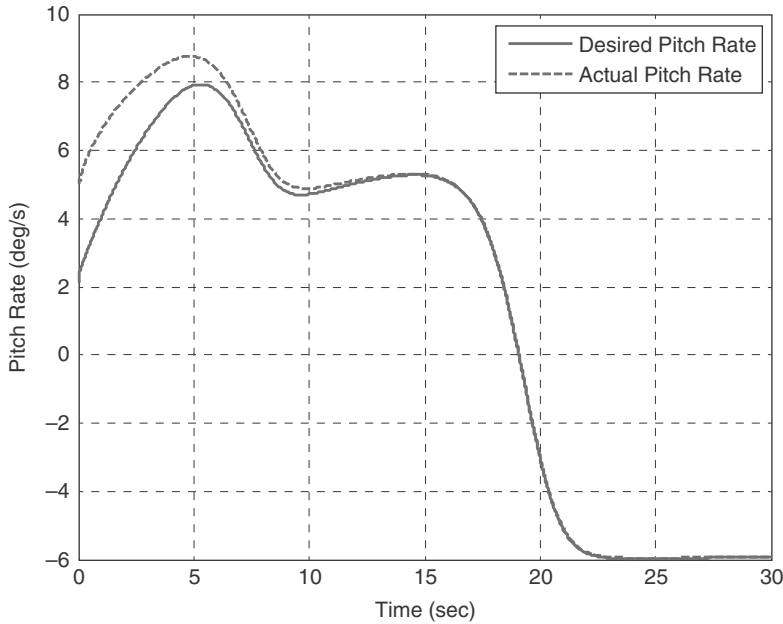


FIG. 11 Histories of desired and actual pitch rate.

The desired control U_d is computed as follows: the control objective is to adjust U_k so that x_{1k} tracks commanded reference input $r(t)$ in Eq. (42). With $x_{1d}(k) = r(t_k)$, it can be computed that $x_{2d}(k) = \dot{x}_{1d}(k) + 1.019x_{1d}(k) - d_1 = \dot{r}(t_k) + 1.019r(t_k) - d_1$ from the first equation of (43). Here $\dot{x}_{2d}(k)$ is computed numerically by $\dot{x}_{2d}(k) = [x_{2d}(k) - x_{2d}(k-1)]/\Delta t$. Then the desired control $U_d(k)$ can be obtained by the second equation of (43). Note that the uncertainties d are unknown, and so in the calculation of the desired control d are approximated by the output of the neural network, which calculates it online.

In J-SNAC synthesis, the input to the critic network is $E_k = X_k - X_{d_k}$, and therefore the cost J_k is a function of the network weights \hat{W}^k and E_k , which is given by Eq. (A2) as $J_k = \hat{W}^k \Phi(E_k)$. In this problem, the network weights were initialized at zero. The basis functions $\Phi(E)$ are selected as $[E_1, E_2, E_1^2, E_2^2, E_1 E_2]^T$ and m ; the number of sets in Eq. (A10) is selected as 100.

2. UNCERTAINTY ESTIMATION

In this example, the uncertainty neural network structure is $\hat{d}_i = W_i^T \varphi$, $i = 1, 2$. Here \hat{d}_1 is to approximate $d(\alpha)$, \hat{d}_2 is to approximate $f(\alpha, \delta_e)$. The design parameter K is selected as 10. W_1 and W_2 are both initialized as 9×1

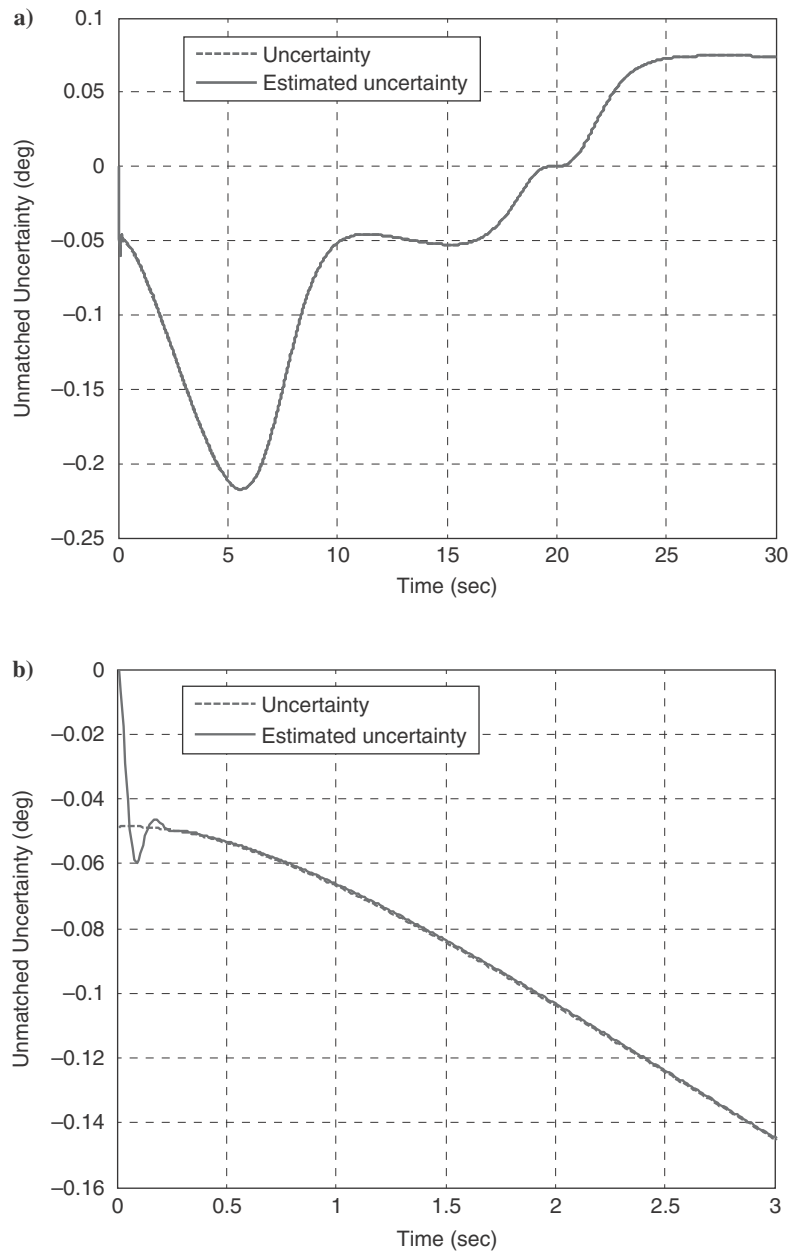


FIG. 12 Histories of true and estimated uncertainties and zoomed at the beginning.

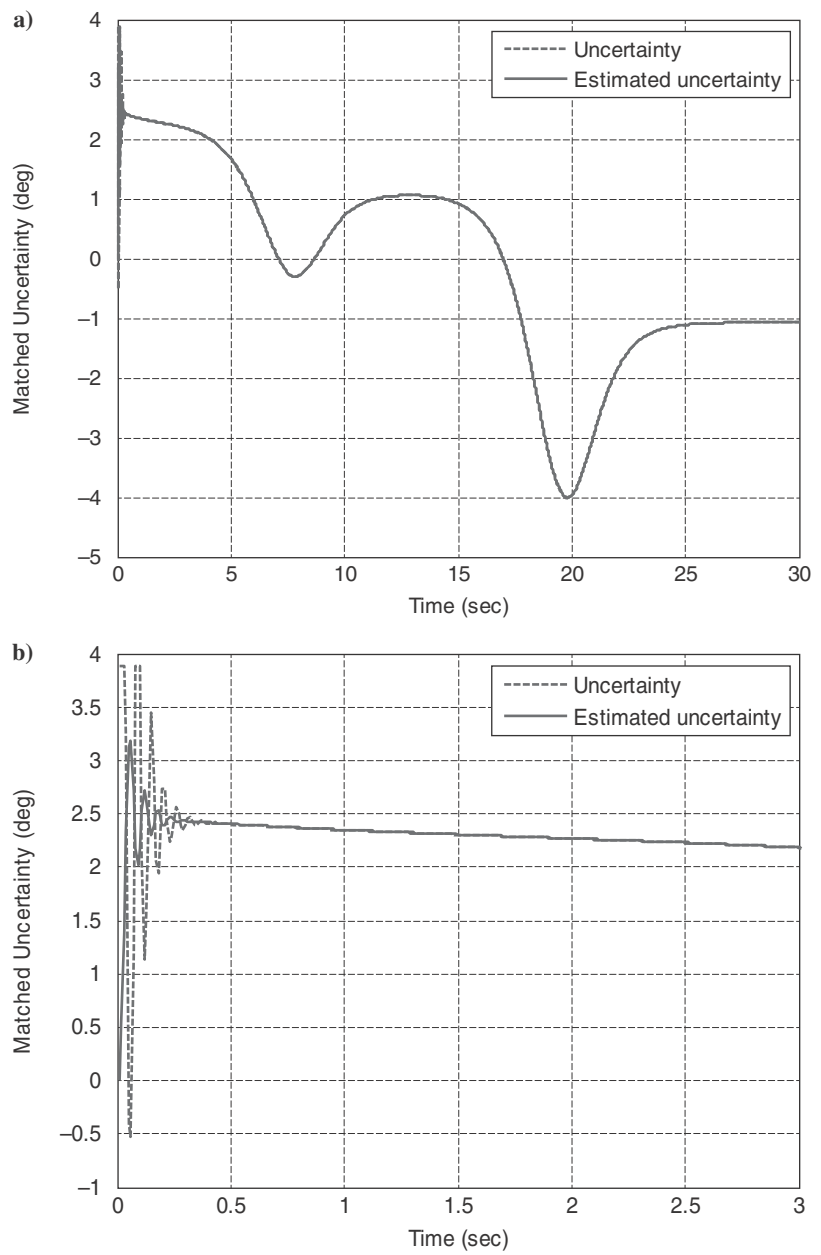


FIG. 13 Histories of true and estimated uncertainties and zoomed at the beginning.

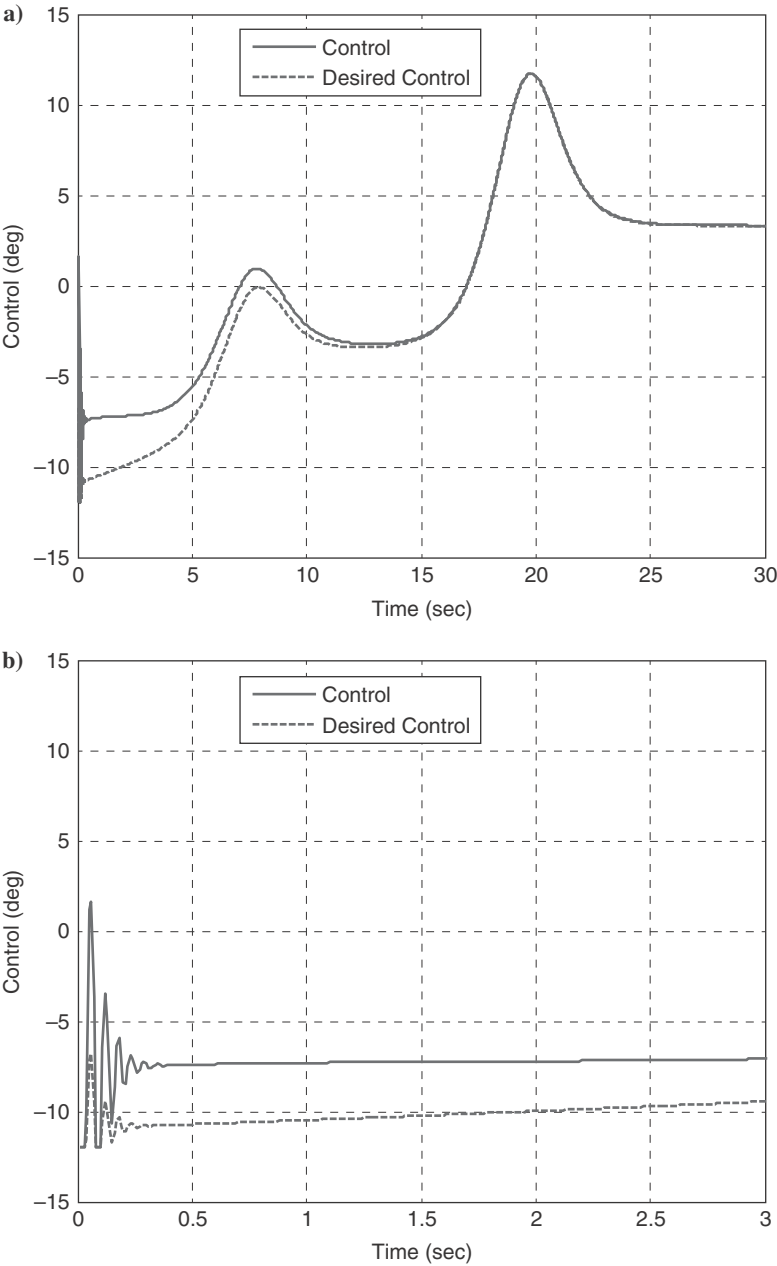


FIG. 14 Histories of constrained control and desired control and zoomed at the beginning.

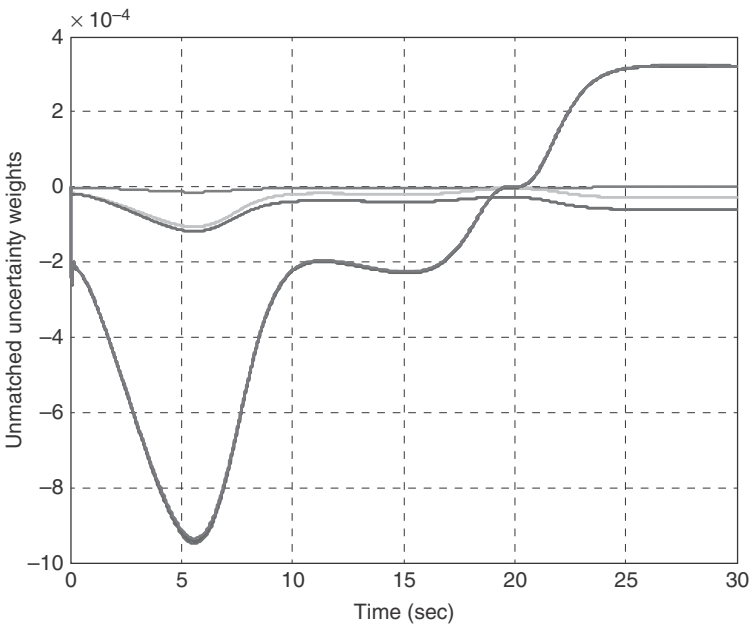


FIG. 15 Weights histories of the unmatched uncertainty estimation.

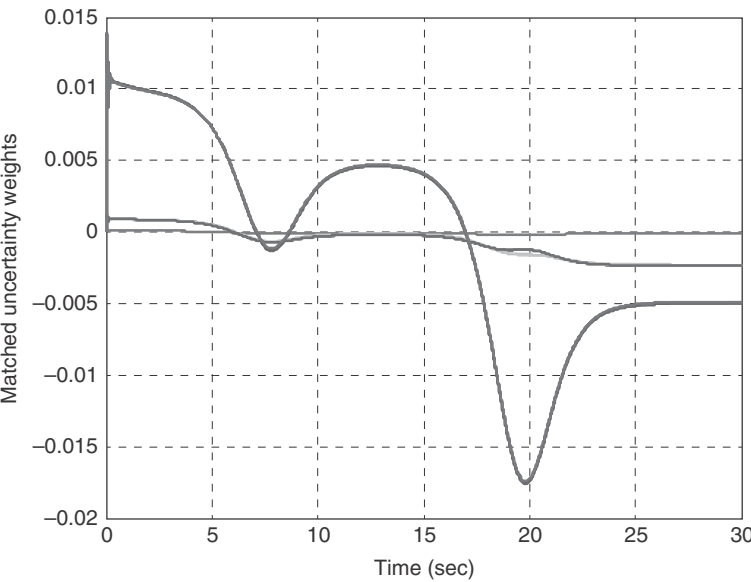


FIG. 16 Weights histories of the matched uncertainty estimation.

zero vectors. The basis functions are selected as $\varphi = \text{kron}(c_1, c_2)$, where $c_1 = [1 \sin(\alpha) \cos(\alpha)]^T$, $c_2 = [1 \sin(q) \cos(q)]^T$, and kron denotes the Kronecker product.

The discretized state equations and the control are the same as Eqs. (39) and (24), respectively. The costate equation though changes to

$$\lambda_k = \Delta t Q_W(X_k - X_{d_k}) + [\partial F'_k / \partial X_k]^T \lambda_{k+1} \quad (45)$$

where F'_k represents the expression on the right-hand side of Eq. (39), which also accounts for the uncertainty. During each iteration of the simulation, the critic network is updated. Because d_k is unknown, \hat{d}_k , the output of the online neural network, is used. The online training is carried out at every instant k by using E_k as the input to the cost network and obtaining the new target cost J'_k as the output.

3. ANALYSIS OF RESULTS

Actual and desired angle of attack and pitch rate histories are shown in Figs. 10 and 11, respectively. There are initial errors because the initial angle of attack is far from the desired. However, the errors between the desired angle attack decrease quickly and the tracking performance is very good after the transient stage. A similar pattern can be observed in the pitch rate history too.

True and estimated uncertainty histories are shown in Figs. 12 and 13. After large initial fluctuations, it can be seen that the estimated uncertainties nicely and quickly track the true uncertainties. Oscillations were observed at the beginning, and so zoomed plots of that region have been provided also. The oscillations occur due to large learning rates used in the uncertainty estimation.

Figure 14 shows the history of the constrained control, which is within the control constraint ($U_{\max} = 12$ deg). Because the initial angle-of-attack deviation is large, the initial actual control is at the maximum level. However, once the uncertainties are computed accurately (around 15 s), it can be seen that the actual control starts overlapping the desired control values.

Weights histories of the uncertainty NNs are shown in Figs. 15 and 16, which also show quick convergence.

VII. CONCLUSION

In this chapter, an online cost-function-based single network adaptive critic (J-SNAC) technique has been presented to solve nonlinear constrained control problems with model uncertainties that are common in the field of aerospace engineering. A nonquadratic cost function was used to incorporate the control constraints. Necessary equations for optimal control were derived, and an algorithm to solve the constrained-control problem with J-SNAC was developed.

Extensions to constrained optimal control problems in the presence of uncertainties were also considered. An aircraft control problem and a spacecraft control problem were used to illustrate the working of the proposed technique. From the simulation results, the proposed method seems to have a good potential for many aerospace applications.

APPENDIX: J-SNAC WEIGHT UPDATE LAW

This appendix is derived from the convergence proof by Al-Tamimi et al. [9]. The difference is that the action network is eliminated in this study. Consider a discrete nonlinear control-affine system

$$X_{k+1} = f(X_k) + BU_k \quad (A1)$$

where X_k is an $n \times 1$ vector, U_k is an $m \times 1$ vector, and $f()$ can be a nonlinear function of the states. J_k is a function of the network weights \hat{W}^k and the basis functions that are functions of the states X_k as

$$J_k = \hat{W}^{kT} \Phi(X_k) \quad (A2)$$

$$\lambda_k = \partial \left[\hat{W}^{kT} \Phi(X_k) \right] / \partial X_k \quad (A3)$$

where it is assumed that X_k stay within a compact domain and $X_k \subset \Omega_c$ so that $\partial \Phi(X_k) / \partial X_k$ is bounded.

In i th iteration, the control is computed by Eq. (8), and the cost relationship at stages k and $k+1$ is given by

$$J_k^i = \Psi_k^i + J_{k+1}^i \quad (A4)$$

where Ψ_k^i represents the “utility function” at time step k .

By using Eq. (A3), if a nonquadratic cost function (11) is selected, the expression for control at the i th iteration is

$$U_k^i = \mu \left(-R^{-1} B^T \{ [\partial f(X_k) / \partial X_k]^T \}^{-1} \left[\frac{\partial \hat{W}^{iT} \Phi(X_k)}{\partial X_k} - QX_k \right] \right) \quad (A5)$$

Substituting Eq. (A2) into Eq. (A4), we obtain

$$\hat{W}^{i+1T} \Phi(X_k) = \Psi_k^i + \hat{W}^{iT} \Phi(X_{k+1}) \quad (A6)$$

Equation (A6) is linear in \hat{W}^{i+1} with m unknowns, where m is number of the basis function elements of vector $\Phi(X_k)$. Taking the transpose of Eq. (A6) and selecting m sets of states X_k called $X_k^{(1)}$ to $X_k^{(m)}$, it ends up with m equations and m unknowns:

$$\begin{cases} \Phi(X_k^{(1)})^T \hat{W}^{i+1} = \Psi_k^{i,1} + \Phi(X_{k+1}^{(1)})^T \hat{W}^i \\ \vdots \\ \Phi(X_k^{(m)})^T \hat{W}^{i+1} = \Psi_k^{i,m} + \Phi(X_{k+1}^{(m)})^T \hat{W}^i \end{cases} \quad (\text{A7})$$

Now, the controller iteration and the state update equation with the set of states $j = 1, 2, \dots, m$ are given by

$$\begin{aligned} U_k^{i(j)} &\equiv U^i(X_k^{(j)}) \\ &= \mu \left(-R^{-1} B^T \{ [\partial f(X_k) / \partial X_k]^T \}^{-1} \left[\frac{\partial \hat{W}^{iT} \Phi(X_k^{(j)})}{\partial X_k^{(j)}} - Q X_k^{(j)} \right] \right) \end{aligned} \quad (\text{A8})$$

$$X_{k+1}^{(j)} = f(X_k^{(j)}) + B U_k^{i(j)} \quad (\text{A9})$$

Equations (A7) can be rewritten as

$$\Phi(X_k) \hat{W}^{i+1} = \text{RHS}(X_k, \hat{W}^i) \quad (\text{A10})$$

where the right-hand (RHS) is the $m \times 1$ vector composed of the right-hand side of Eq. (A7) and the $m \times m$ matrix $\Phi(X_k)$ is given by

$$\Phi(X_k) = \begin{bmatrix} \Phi(X_k^{(1)})^T \\ \vdots \\ \Phi(X_k^{(m)})^T \end{bmatrix}, \quad X_k = \begin{bmatrix} X_k^{(1)T} \\ \vdots \\ X_k^{(m)T} \end{bmatrix} \quad (\text{A11})$$

By using Eq. (A11) in Eq. (A10), a recursive relationship for the network weights is given as

$$\hat{W}^{i+1} = \Phi(X_k)^{-1} \text{RHS}(X_k, \hat{W}^i) \quad (\text{A12})$$

For the inverse $\Phi(X_k)^{-1}$ to exist, $X_k^{(j)}$ should not be identical, and the elements of vector $\phi(X_k)$ should be linearly independent. One can select more than m minimum required sets of states and formulate a recursive relationship for the overdefined system of equations. In fact, it is advisable to use a large number of

data sets because it will better enable the network to capture the long-term behavior or the system evolution for many different initial conditions and may help with faster convergence. In this case, the unique solution of the least-squares minimization problem is simply

$$\hat{W}^{i+1} = [\Phi(X_k)^T \Phi(X_k)]^{-1} \Phi(X_k)^T \text{RHS}(X_k, \hat{W}^i) \quad (\text{A13})$$

The convergence of the proposed algorithm can also be derived from [9] with some modifications. The Lyapunov function can be selected as $J_\infty = \frac{1}{2} X_k^T Q_W X_k + \int \{\mu^{-1}[v_\infty(x_k)]\}^T R_W dv(x_k) + J_\infty$.

ACKNOWLEDGMENTS

Support for this study from NASA under Grant No. ARMD NRA NNH07ZEA001N-IRAC1 and the National Science Foundation (NSF) are gratefully acknowledged. The views of the authors do not necessarily represent the views of the NSF and NASA.

REFERENCES

- [1] Lewis, F. L., *Applied Optimal Control and Estimation*, Prentice-Hall, Upper Saddle River, NJ, 1992.
- [2] Bryson, A. E., and Ho, Y. C., *Applied Optimal Control*, Taylor and Francis, Washington, D.C., 1975.
- [3] Werbos, P. J., "Approximate Dynamic Programming for Real-Time Control and Neural Modeling," *Handbook of Intelligent Control—Neural, Fuzzy, and Adaptive Approaches*, Van Nostrand Reinhold, New York, 1992, pp. 493–525.
- [4] Barto, A. G., "Connectionist Learning for Control," *Neural Networks for Control*, MIT Press, Cambridge, MA, 1991.
- [5] Barto, A. G., and Dieterich, T., "Reinforcement Learning and Its Relation to Supervised Learning," *Handbook of Learning and Approximate Dynamic Programming*, Wiley-IEEE Press, New York, Aug. 2004.
- [6] Powell, W., and Van Roy, B., "ADP for High-Dimensional Resource Allocation Problems," *Handbook of Learning and Approximate Dynamic Programming*, Wiley-IEEE Press, New York, 2004.
- [7] Bertsekas, D. P., and Tsitsiklis, J. N., *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996.
- [8] Si, J., Barto, P. A. G., and Wunsch, W. B. D. (eds.), *Handbook of Learning and Approximate Dynamic Programming*, IEEE Press Series on Computational Intelligence, Wiley-IEEE Press, New York, 2004.
- [9] Al-Tamimi, A., Lewis, F. L., and Abu-Khalaf, M., "Discrete-Time Nonlinear HJB Solution Using Approximate Dynamic Programming: Convergence Proof," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 38, No. 4, 2008, pp. 943–949.

- [10] Balakrishnan, S. N., Ding, J., and Lewis, F. L., "Issues on Stability of ADP Feedback Controllers for Dynamical Systems," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 38, No. 4, 2008, pp. 913–917.
- [11] Li, B., and Si, J., "Robust Dynamic Programming for Discounted Infinite-Horizon Markov Decision Processes with Uncertain Stationary Transition Matrices," *Proceedings of the IEEE International Symposium Approximate Dynamic Programming and Reinforcement Learning*, 2007, pp. 96–102.
- [12] Werbos, P. J., "Using ADP to Understand and Replicate Brain Intelligence: the Next Level Design," *Proceedings of the IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007, pp. 209–216.
- [13] Werbos, P. J., "Neural Networks for Control and System Identification," *Proceedings of the 28th IEEE Conference on Decision and Control*, Vol. 1, 1989, pp. 260–265.
- [14] Werbos, P. J., "A Menu of Designs for Reinforcement Learning Over Time," *Neural Networks for Control*, edited by W. T. Miller, R. S. Sutton, and P. J. Werbos, MIT Press, Cambridge, MA, 1991, pp. 67–95.
- [15] Zheng, C., and Jagannathan, S., "Generalized Hamilton-Jacobi-Bellman Formulation-Based Neural Network Control of Affine Nonlinear Discrete-Time Systems," *IEEE Transactions on Neural Networks*, Vol. 19, No. 1, Jan. 2008, pp. 90–106.
- [16] Yang, Q., and Jagannathan, S., "Adaptive Critic Neural Network Force Controller for Atomic Force Microscope-Based Nanomanipulation," *Proceedings of the 2006 IEEE International Symposium on Intelligent Control*, 2006, pp. 464–469.
- [17] He, P., and Jagannathan, S., "Reinforcement Learning Neural-Network-Based Controller for Nonlinear Discrete-Time Systems with Input Constraints," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 37, No. 2, Apr. 2007, pp. 425–436.
- [18] Al-Tamimi, A., Lewis, F. L., and Wang, Y., "Model-Free H-Infinity Load-Frequency Controller Design for Power Systems," *Proceedings of the 22nd IEEE International Symposium on Intelligent Control*, 2007, pp. 118–125.
- [19] Vrabie, D., Pastravanu, O., and Lewis, F. L., "Policy Iteration for Continuous-Time Systems with Unknown Internal Dynamics," *Proceedings of the 15th Mediterranean Conference on Control and Automation*, 2007, pp. 1–6.
- [20] Shih, P., Kaul, B., Jagannathan, S., and Drallmeier, J., "Near Optimal Output-Feedback Control of Nonlinear Discrete-Time Systems in Nonstrict Feedback Form with Application to Engines," *Proceedings of the International Joint Conference on Neural Networks*, 2007, pp. 396–401.
- [21] Balakrishnan, S. N., and Biega, V., "Adaptive-Critic Based Neural Networks for Aircraft Optimal Control," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 4, 1996, pp. 893–898.
- [22] Prokhorov, D., and Wunsch, D., II, "Adaptive Critic Designs," *IEEE Transactions on Neural Networks*, Vol. 8, No. 4, 1997, pp. 997–1007.
- [23] Venayagamoorthy, G., Harley, R., and Wunsch, D., "Dual Heuristic Programming Excitation Neurocontrol for Generators in a Multimachine Power System," *IEEE Transactions on Industry Applications*, Vol. 39, No. 2, March/April 2003, pp. 382–384.
- [24] Ferrari, S., and Stengel, R., "An Adaptive Critic Global Controller," *Proceedings of the American Control Conference*, 2002, pp. 2665–2670.

- [25] Lendaris, G., Schultz, L., and Shannon, T., "Adaptive Critic Design for Intelligent Steering and Speed Control of a 2-Axle Vehicle," *Proceedings of the International Joint Conference on Neural Networks*, 2000.
- [26] Hanselmann, T., Noakes, L., and Zaknich, A., "Continuous Time Adaptive Critics," *IEEE Transactions on Neural Networks*, Vol. 18, No. 3, 2007, pp. 631–647.
- [27] Padhi, R., and Balakrishnan, S. N., "Optimal Beaver Population Management Using Reduced Order Distributed Parameter Model and Single Network Adaptive Critics," *Proceedings of the American Control Conference*, 2004, pp. 1598–1603.
- [28] Padhi, R., Unnikrishnan, N., and Balakrishnan, S. N., "Optimal Control Synthesis of a Class of Nonlinear Systems Using Single Network Adaptive Critics," *Proceedings of the American Control Conference*, 2004, pp. 1592–1597.
- [29] Yadav, V., Padhi, R., and Balakrishnan, S. N., "Robust/Optimal Temperature Profile Control Using Neural Networks," *Proceedings of the IEEE International Conference on Control Applications*, 2006, pp. 3169–3174.
- [30] Yang, L., Si, J., Tsakalis, K. S., and Rodriguez, A. A., "Direct Heuristic Dynamic Programming for Nonlinear Tracking Control with Filtered Tracking Error," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 39, No. 6, 2009, pp. 1617–1622.
- [31] Wang, F., Zhang, H., and Liu, D., "Adaptive Dynamic Programming: An Introduction," *IEEE Computational Intelligence Magazine*, Vol. 4, No. 2, 2011, pp. 39–47.
- [32] Ding, J., and Balakrishnan, S. N., "Approximate Dynamic Programming Solutions with a Single Network Adaptive Critic for a Class of Nonlinear Systems," *Journal of Control Theory and Applications*, Vol. 9, No. 3, 2011, pp. 370–380.
- [33] Seborg, D. E., Edgar, T. E., Mellichamp, D. A., and Doyle, F. J., III, *Process Dynamics and Control*, Wiley, Hoboken, NJ, 2011.
- [34] Ryan, E. P., "Optimal Feedback Control of Saturating Systems," *International Journal of Control*, Vol. 35, No. 3, 1982, pp. 521–534.
- [35] Bernstein, D. S., "Optimal Nonlinear, but Continuous, Feedback Control of Systems with Saturating Actuators," *International Journal of Control*, Vol. 62, No. 5, 1995, pp. 1209–1216.
- [36] Frankena, J. F., and Sivan, R., "A Non-Linear Optimal Control Law for Linear Systems," *International Journal of Control*, Vol. 30, No. 1, 1979, pp. 159–178.
- [37] Dolphus, R. M., and Schmitendorf, W. E., "Stability Analysis for a Class of Linear Controllers Under Control Constraints," *Proceedings of the 30th IEEE Conference on Decision and Control*, Vol. 1, IEEE, Piscataway, NJ, 1991, pp. 77–80.
- [38] Bernstein, D. S., "Optimal Nonlinear, But Continuous, Feedback Control of Systems with Saturating Actuators," *Proceedings of the 32nd Conference on Decision and Control*, 1993, pp. 2533–2537.
- [39] Yang, R., and Wu, C., "Neural Networks for Exact Solution of Constrained Optimal Control Problems," *Proceedings of the American Control Conference*, 1994, pp. 1379–1383.
- [40] Lyshevski, S., "Robust Nonlinear Control of Uncertain Systems with State and Control Constraints," *Proceedings of the 34th Conference on Decision and Control*, 1995, pp. 1670–1675.
- [41] Lyshevski, S., "Control of Linear Dynamic Systems with Constraints: Optimization Issues and Applications of Nonquadratic Functionals," *Proceedings of the 35th Conference on Decision and Control*, 1996, pp. 3206–3211.

- [42] Lyshevski, S., "Optimal Tracking Control of Nonlinear Dynamic Systems with Control Bounds," *Proceedings of the 38th Conference on Decision and Control*, 1999, pp. 4810–4815.
- [43] Adhyaru, D. M., and Kar, I. N., "Constrained Optimal Control of Bilinear Systems Using Neural Network Based HJB Solution," *2008 International Joint Conference on Neural Networks*, 2008, pp. 4137–4142.
- [44] Cheng, T., and Lewis, F. L., "Fixed-Final Time Constrained Optimal Control of Nonlinear Systems Using Neural Network HJB Approach," *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 3016–3021.
- [45] Gupta, S. K., *Numerical Methods for Engineers*, Wiley Eastern Ltd. and New Age International Ltd., New York, 1995.
- [46] Beard, R. W., "Improving the Closed-Loop Performance of Nonlinear Systems," Ph.D. Dissertation, Rensselaer Polytechnic Inst., Troy, NY, Oct. 1995.
- [47] Lyshevski, S. E., *Control Systems Theory with Engineering Applications*, Birkhäuser. Cambridge, MA, 2001.
- [48] Khalil, H. K., *Nonlinear Systems*, 3rd ed., Prentice–Hall, Upper Saddle River, NJ, 2002.
- [49] Bishop, C., *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, England, U.K., 1995.
- [50] Hagan, M. T., Demuth, H. B., and Beale, M., *Neural Network Design*, PWS Publishing, Boston, 1996.
- [51] Haykin, S., *Neural Networks: A Comprehensive Foundation*, Prentice–Hall, Upper Saddle River, NJ, 1999.
- [52] Padhi, R., Unnikrishnan, N., and Balakrishnan, S. N., "Model-Following Neuro-Adaptive Control Design for Non-Square, Non-Affine Nonlinear Systems," *Control Theory and Applications, IET*, Vol. 1, No. 6, 2007, pp. 1650–1661.
- [53] Unnikrishnan, N., "Neural Network Based Robust Nonlinear Control," Ph.D. Dissertation, Missouri Univ. of Science and Technology, Rolla, MO, 2006.
- [54] Unnikrishnan, N., Balakrishnan, S. N., and Padhi, R. "Dynamic Re-optimization of a Spacecraft Attitude Controller in the Presence of Uncertainties," *Proceedings of the 2006 IEEE International Symposium on Intelligent Control*, 2006, pp. 452–457.
- [55] Slotine, J.-J. E., and Li, W., *Applied Nonlinear Control*, Prentice–Hall, Upper Saddle River, NJ, 1991, Chap. 9.
- [56] Young, A., Cao, C., Patel, V., and Hovakimyan, N., "Adaptive Control Design Methodology for Nonlinear-in-Control Systems in Aircraft Applications," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 6, 2007, pp. 1770–1782.

Modified Reference Model MRAC (M-MRAC): An Application to a Generic Transport Aircraft

Vahram Stepanyan* and Kalmanje Krishnakumar[†]

NASA Ames Research Center, Moffett Field, California 94035

I. INTRODUCTION

THE asymptotic behavior of adaptive systems has been a well-researched topic during the last couple of decades, and it is well known that asymptotic tracking can be achieved using the Lyapunov redesign method. However, the transient behavior of the input and output signals can be oscillatory with big excursions [1]. There has been a great deal of effort to modify the control architecture and the adaptive laws from the perspective of improving the transient behavior of the tracking error. The majority of these efforts led to nonadaptive high gain feedback [2–4], switching control law [5, 6], or to a parameter-dependent persistent excitation condition [7].

In general, increasing the adaptation rate while reducing the tracking error magnitude (for example, see [8]), generates high-frequency oscillations and big overshoot in the control signal leading to possible actuator failures or excitation of unmodeled dynamics, which in turn can drive the overall system to instability. This shortcoming is common for the majority of existing adaptive control methods. Recently, some results that explicitly address the input signal transient behavior, which is very important from the point of view of the performance specifications of closed-loop systems, have been appearing in the control community.

The first contribution to transient analysis of the adaptive control signal can be found in [9], where it is shown that the bound on the control signal is proportional to the square root of the adaptation rate. This result is conservative, but it reflects the general observations about the control signal behavior of the MRAC system. In [10], a new adaptive control architecture, called L_1 adaptive control, has been introduced, which can achieve arbitrarily close tracking of a given reference command both in transient and steady state by increasing the

*Senior Scientist, Mission Critical Technologies, Inc.; vahram.stepanyan@nasa.gov.

[†]Autonomous Systems and Robotics Tech Area Lead, Intelligent Systems Division; kalmanje.krishnakumar@nasa.gov.

adaptation gain. The low-pass filter introduced in the control channel prevents high-frequency oscillations in the control signal, which closely follows the ideal one. However, the approach loses the possibility to have a real reference model to track, for which the control metrics can be specified.

An alternative approach is proposed in [11] for a class of multi-input multi-output uncertain nonlinear systems subject to external disturbances to track a given reference model. Along with the tracking error, it also uses its integral to guarantee asymptotic tracking and transient performance of both input and output signals. The control architecture internally generates a low-pass filter, which eliminates high-frequency oscillations for large adaptation rates.

In this chapter, we take a somewhat different approach. Instead of modifying the control architecture or the adaptive laws, we modify the reference model by feeding back the tracking error signal. This approach, called the

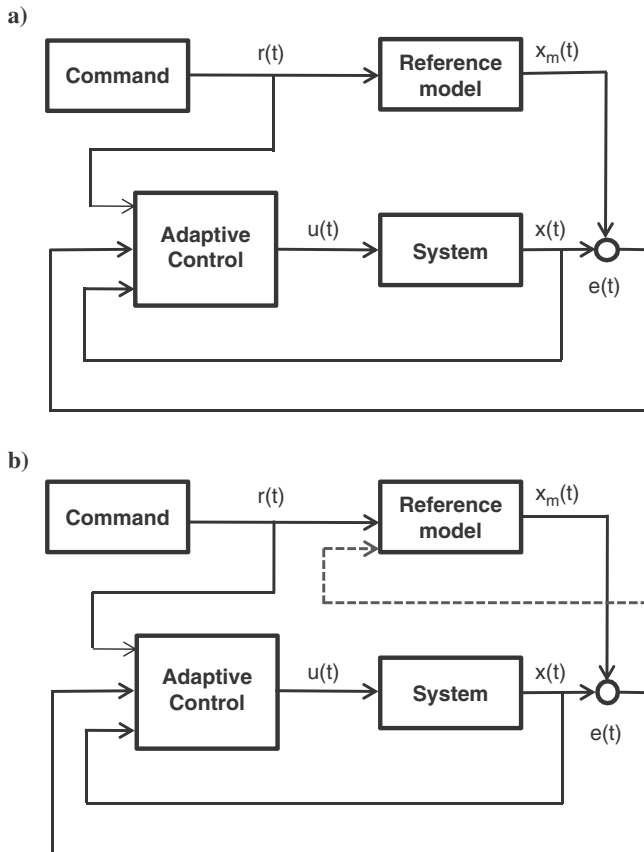


FIG. 1 MRAC and M-MRAC control systems: a) MRAC and b) M-MRAC.

modified-reference-model MRAC or M-MRAC in short, is motivated by the fact that the initially large error in the control gains generates large transient excursions both in the system's input and output signals. Moreover, the output excursion, in turn, generates oscillations in the control signal. The idea is to prevent the system's attempt to aggressively maneuver toward the reference model by modifying the reference model by a term proportional to the tracking error. As the tracking error approaches zero, the modified reference model approaches its original form. Therefore, the system asymptotically tracks not only the modified reference model, but also the original reference model. In addition, the error feedback term determines the damping in the control signal, an increase of which makes it possible to increase the learning rate for better transient performance without generating oscillations in the adaptive system. Meanwhile, excessively large error feedback gain may increase the overshoot with respect to the original reference model. Therefore, both parameters need to be increased simultaneously for better transient performance, which can be made arbitrarily close to the ideal performance of the known system. A design guideline is provided for the selection of the feedback gain relative to the adaptation rate. The proposed adaptive control method has uniform performance for all reference commands and initial conditions without the need for retuning. Figure 1 emphasizes the difference between MRAC and M-MRAC architectures.

II. PRELIMINARIES

In this section, we derive a minimal upper bound for the solution of a second-order linear-time-variant system. Let

$$\ddot{x}(t) + 2a\dot{x}(t) + \gamma k(t)x(t) = b_1\dot{f}(t) + b_2f(t) \quad x(0) = x_0, \dot{x}(0) = \dot{x}_0 \quad (1)$$

where $\gamma > 0$ is a constant parameter and $k(t)$ is continuous with $k^* \geq k(t) \geq k_*$ for some positive constants $k^* > k_*$ and has a bounded derivative almost everywhere. The function $f(t)$ is assumed to be piecewise continuous and bounded. Equation (1) can be written in the matrix form as

$$\dot{\mathbf{z}}(t) = \mathbf{A}\mathbf{z}(t) + \mathbf{B}f(t) \quad (2)$$

where

$$\mathbf{z}(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) - b_1f(t) \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\gamma k(t) & -2a \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} b_1 \\ b_2 - 2ab_1 \end{bmatrix}$$

We are interested in minimizing the upper bound on $\mathbf{z}(t)$ by the choice of the parameter a . To this end, we introduce notations $\omega^2 = \gamma k_0$, $k_0 = (k^* + k_*)/2$, $a = \zeta\omega$ and represent the system (2) in the following equivalent form:

$$\dot{\mathbf{z}}(t) = \mathbf{D}\mathbf{z}(t) + \mathbf{B}f(t) + \mathbf{C}[\omega^2 - \gamma k(t)]x(t) \quad (3)$$

where

$$D = \begin{bmatrix} 0 & 1 \\ -\omega^2 & -2\zeta\omega \end{bmatrix}, \quad C = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

For the convenience of derivations, we decompose $\mathbf{z}(t)$ into initial response $\mathbf{z}^i(t)$ of the homogeneous system

$$\dot{\mathbf{z}}^i(t) = A\mathbf{z}^i(t) \quad (4)$$

with the initial condition $\mathbf{z}_0 = [x_0 \quad \dot{x}_0 - b_1 f(0)]^\top$ and forced response $\mathbf{z}^f(t)$ of the system (3) with zero initial conditions, which can be represented in the equivalent integral form

$$\mathbf{z}^f(t) = \int_0^t G(t-\tau)Bf(\tau) d\tau + \int_0^t G(t-\tau)C[\omega^2 - \gamma k(t)]x^f(\tau) d\tau \quad (5)$$

Here, $G(t) = e^{Dt}$, which can be computed by direct integration and has the form [12]

$$G(t) = \begin{cases} e^{-\zeta\omega t} \begin{bmatrix} \cos(\delta t) + \frac{\zeta\omega}{\delta} \sin(\delta t) & \frac{1}{\delta} \sin(\delta t) \\ -\frac{\omega^2}{\delta} \sin(\delta t) & \cos(\delta t) - \frac{\zeta\omega}{\delta} \sin(\delta t) \end{bmatrix}, & \zeta < 1 \\ e^{-\zeta\omega t} \begin{bmatrix} 1 + \omega t & t \\ -\omega^2 t & 1 - \omega t \end{bmatrix}, & \zeta = 1 \\ e^{-\zeta\omega t} \begin{bmatrix} \cosh(\delta t) + \frac{\zeta\omega}{\delta} \sinh(\delta t) & \frac{1}{\delta} \sinh(\delta t) \\ -\frac{\omega^2}{\delta} \sinh(\delta t) & \cosh(\delta t) - \frac{\zeta\omega}{\delta} \sinh(\delta t) \end{bmatrix}, & \zeta > 1 \end{cases} \quad (6)$$

where $\delta = \omega\sqrt{1 - \zeta^2}$. To minimize the bound on $\mathbf{z}^f(t)$, we compute the \mathcal{L}_1 norm of matrix $G(t)$ component wise. For $g_{12}(t)$, we obtain

$$\|g_{12}(t)\|_{\mathcal{L}_1} = \begin{cases} \frac{1}{\omega^2} \left\{ \left[\exp\left(\frac{\zeta\omega\pi}{\delta}\right) + 1 \right] / \left[\exp\left(\frac{\zeta\omega\pi}{\delta}\right) - 1 \right] \right\}, & \zeta < 1 \\ \frac{1}{\omega^2}, & \zeta \geq 1 \end{cases} \quad (7)$$

where $0 < \varphi = \tan^{-1}\left(\frac{\delta}{\zeta\omega}\right) < \frac{\pi}{2}$. Obviously, $\|g_{12}(t)\|_{\mathcal{L}_1}$ reaches its minimum of $\frac{1}{\omega^2}$ for all $\zeta \geq 1$. Because $g_{12}(t) = -\omega^2 g_{21}(t)$, it follows that $\|g_{21}(t)\|_{\mathcal{L}_1}$ attains a

unity minimum value for all $\zeta \geq 1$. On the other hand, the \mathcal{L}_1 norm of $g_{11}(t)$ is computed to be

$$\|g_{11}(t)\|_{\mathcal{L}_1} = \begin{cases} \frac{2}{\omega} \left\{ \zeta + \left[\frac{\exp\left(\frac{\zeta\omega\varphi}{\delta}\right)}{\exp\left(\frac{\zeta\omega\pi}{\delta}\right) - 1} \right] \right\}, & \zeta < 1 \\ \frac{2\zeta}{\omega}, & \zeta \geq 1 \end{cases} \quad (8)$$

It can be shown that the minimum of $\|g_{11}(t)\|_{\mathcal{L}_1}$ is reached at some $\zeta^* < 1$. The analytical computations are somewhat involved, but numerical computations result in $\zeta^* = 0.66$ with the minimum value of $2c/\omega$, where $c = 0.8026$. Finally, computations of $\|g_{22}(t)\|_{\mathcal{L}_1}$ result in

$$\|g_{22}(t)\|_{\mathcal{L}_1} = \begin{cases} \frac{2}{\omega} \left\{ \zeta + \left[\exp\left(\frac{-\zeta\omega\varphi}{\delta}\right) / \exp\left(\frac{\zeta\omega\pi}{\delta}\right) - 1 \right] \right\}, & \zeta < 1 \\ \frac{2}{\omega e}, & \zeta = 1 \\ \frac{2}{\omega} \frac{\zeta}{\sqrt{\zeta^2 - 1}} \left(\frac{\zeta + \sqrt{\zeta^2 - 1}}{\zeta - \sqrt{\zeta^2 - 1}} \right)^{-\frac{\zeta}{2\sqrt{\zeta^2 - 1}}}, & \zeta > 1 \end{cases} \quad (9)$$

It can be shown that $\|g_{22}(t)\|_{\mathcal{L}_1}$ is a decreasing function in ζ and $\|g_{22}(t)\|_{\mathcal{L}_1} \rightarrow 0$ as $\zeta \rightarrow \infty$. Because there is no common minimum point for the entries of $G(t)$, one can use a “suboptimal” value $\zeta = 1$. In this case $\|g_{11}(t)\|_{\mathcal{L}_1} = 2/\omega^2$, $\|g_{12}(t)\|_{\mathcal{L}_1} = 1/\omega^2$, $\|g_{21}(t)\|_{\mathcal{L}_1} = 1$, and $\|g_{22}(t)\|_{\mathcal{L}_1} = 2/\omega e$. We notice that selecting a larger value of ζ while leaving $\|g_{12}(t)\|_{\mathcal{L}_1}$ and $\|g_{21}(t)\|_{\mathcal{L}_1}$ intact increases $\|g_{11}(t)\|_{\mathcal{L}_1}$ and decreases $\|g_{22}(t)\|_{\mathcal{L}_1}$, both proportional to $1/\omega$. Therefore, we can select any $\zeta \geq 1$ with $\|g_{11}(t)\|_{\mathcal{L}_1} = c_1/\omega$ and $\|g_{22}(t)\|_{\mathcal{L}_1} = c_2/\omega$, where $c_1 \geq 2$ and $\frac{2}{e} \geq c_2 > 0$ are constants defined by the selected ζ and independent of ω .

Next we compute \mathcal{L}_∞ bound on the components of vector $\mathbf{z}^f(t)$. For the first component we have

$$\begin{aligned} x^f(t) &= \int_0^t [b_1 g_{11}(t - \tau) + (b_2 - 2ab_1) g_{12}(t - \tau)] f(\tau) d\tau \\ &\quad + \int_0^t g_{12}(t - \tau) [\omega^2 - \gamma k(t)] x^f(\tau) d\tau \end{aligned} \quad (10)$$

Because $\|\omega^2 - \gamma k(t)\|_{\mathcal{L}_\infty} = \omega^2 - \gamma k_* = \gamma \frac{k^* - k_*}{2}$, we obtain (see [13], p. 199 for details)

$$\begin{aligned} \|x^f(t)\|_{\mathcal{L}_\infty} &\leq \|f(t)\|_{\mathcal{L}_\infty} [|b_1| \|g_{11}(t)\|_{\mathcal{L}_1} + |b_2 - 2ab_1| \|g_{12}(t)\|_{\mathcal{L}_1}] \\ &\quad + \gamma \frac{k^* - k_*}{2} \|x^f(t)\|_{\mathcal{L}_\infty} \|g_{12}(t)\|_{\mathcal{L}_1} \end{aligned} \quad (11)$$

Substituting the \mathcal{L}_1 norm values and solving the resulting inequality for $\|x^f(t)\|_{\mathcal{L}_\infty}$, we obtain

$$\left(1 - \frac{k^* - k_*}{2k_0}\right) \|x^f(t)\|_{\mathcal{L}_\infty} \leq \left[\frac{c_1|b_1|}{\omega} + \frac{|b_2 - 2ab_1|}{\omega^2}\right] \|f(t)\|_{\mathcal{L}_\infty} \quad (12)$$

which results in

$$\|x^f(t)\|_{\mathcal{L}_\infty} \leq \left[\frac{c_1|b_1|\sqrt{k_0}}{k_*\sqrt{\gamma}} + \frac{|b_2 - 2ab_1|}{k_*\gamma}\right] \|f(t)\|_{\mathcal{L}_\infty} \quad (13)$$

For the second component $z_2^f(t) = \dot{x}^f(t) - b_1f(t)$, we have

$$\begin{aligned} z_2^f(t) &= \int_0^t [b_1g_{21}(t-\tau) + (b_2 - 2ab_1)g_{22}(t-\tau)]f(\tau) d\tau \\ &\quad + \int_0^t g_{22}(t-\tau)[\omega^2 - \gamma k(t)]x^f(\tau) d\tau \end{aligned} \quad (14)$$

Substituting the \mathcal{L}_1 norm values and using the inequality (13), we obtain

$$\begin{aligned} \|z_2^f(t)\|_{\mathcal{L}_\infty} &\leq \left[|b_1| + \frac{c_2|b_2 - 2ab_1|}{\omega}\right] \|f(t)\|_{\mathcal{L}_\infty} + \frac{c_2\gamma(k^* - k_*)}{2\omega} \|x^f(t)\|_{\mathcal{L}_\infty} \\ &\leq \left\{ |b_1| \left[1 + \frac{c_1c_2(k^* - k_*)}{2k_*}\right] + \frac{c_2|b_2 - 2ab_1|}{\sqrt{k_0}\gamma} \left(1 + \frac{k^* - k_*}{2k_*}\right) \right\} \|f(t)\|_{\mathcal{L}_\infty} \end{aligned} \quad (15)$$

To obtain a bound for $z^i(t)$, we recall that according to Theorem 8.7 [14] the origin of the system (4) is uniformly exponentially stable. Because $\|A(t)\|$ is bounded, $\|\dot{A}(t)\|$ is bounded almost everywhere, and the pointwise eigenvalues of matrix $A(t)$ have negative right-hand sides. Therefore,

$$\|z^i(t)\| \leq c_3 \|z(0)\| e^{-\nu t} \quad (16)$$

for some positive constants c_3 and ν . According to [14] (p. 140), ν can be given by the formula

$$\nu = \frac{\sqrt{\gamma}}{2} \left(\sqrt{\xi k_0} - \sqrt{\xi k_0 - k_*} \right) \quad (17)$$

that is, the rate of decay is proportional to $\sqrt{\gamma}$.

Because Eq. (16) is true for each component of $\mathbf{z}^i(t)$, denoting $c_4 = c_3 \|\mathbf{z}(0)\| = c_3 \sqrt{x_0^2 + [\dot{x}_0^2 + b_1 f(0)]^2}$ and adding the corresponding inequalities, we arrive at

$$\begin{aligned} |z_1(t)| = |x(t)| &\leq c_4 e^{-\nu t} + \left[\frac{c_1 |b_1| \sqrt{k_0}}{k_* \sqrt{\gamma}} + \frac{|b_2 - 2ab_1|}{k_* \gamma} \right] \|f(t)\|_{\mathcal{L}_\infty} \\ |z_2(t)| = |\dot{x}(t) - b_1 f(t)| &\leq c_4 e^{-\nu t} + \left\{ |b_1| \left[1 + \frac{c_1 c_2 (k^* - k_*)}{2k_*} \right] \right. \\ &\quad \left. + \frac{c_2 |b_2 - 2ab_1| \sqrt{k_0}}{\sqrt{\gamma}} \right\} \|f(t)\|_{\mathcal{L}_\infty} \end{aligned} \quad (18)$$

for all $a \geq \sqrt{\gamma k_0}$. From the second inequality in Eq. (18), it follows that

$$|\dot{x}(t)| \leq c_4 e^{-\nu t} + \left\{ |b_1| \left[2 + \frac{c_1 c_2 (k^* - k_*)}{2k_*} \right] + \frac{c_2 |b_2 - 2ab_1| \sqrt{k_0}}{\sqrt{\gamma}} \right\} \|f(t)\|_{\mathcal{L}_\infty} \quad (19)$$

The first inequality in Eq. (18) along with Eq. (19) constitutes the required minimal upper bound for the solution of Eq. (1).

We notice that when \mathbf{x} and \mathbf{f} are q -dimensional vectors in Eq. (1), then $\mathbf{z} = [x_1 \ \dot{x}_1 - b_1 f_1(t) \ \dots \ x_q \ \dot{x}_q - b_1 f_q(t)]^\top$ and the matrices A, B, C, D , and G have repeated block structures. Therefore, Eq. (11) and the upper bound (13) hold for each component $x_i^f(t)$ of vector $\mathbf{x}^f(t)$ with $f(t)$ replaced with $f_i(t)$. Similarly, Eq. (14) and the upper bound (15) hold for each component of vector $\dot{\mathbf{x}}^f(t) - b_1 \mathbf{f}(t)$. On the other hand, the inequality (16) is true for the $2q$ vector $\mathbf{z}^i(t)$; hence, it is true for the vectors $\mathbf{x}^i(t)$ and $\dot{\mathbf{x}}^i(t) - b_1 \mathbf{f}(t)$. That is,

$$\begin{aligned} \|\mathbf{x}^i(t)\| &\leq c_4 e^{-\nu t} \\ \|\dot{\mathbf{x}}^i(t) - b_1 \mathbf{f}(t)\| &\leq c_4 e^{-\nu t} \end{aligned} \quad (20)$$

where now $c_4 = c_3 \sqrt{\|\mathbf{x}_0\|^2 + \|\dot{\mathbf{x}}_0 - b_1 \mathbf{f}(0)\|^2}$. It follows that the upper bounds

$$\begin{aligned} \|\mathbf{x}(t)\| &\leq c_4 e^{-\nu t} + \left[\frac{c_1 |b_1| \sqrt{k_0}}{k_* \sqrt{\gamma}} + \frac{|b_2 - 2ab_1|}{k_* \gamma} \right] \|\mathbf{f}(t)\|_{\mathcal{L}_\infty} \\ \|\dot{\mathbf{x}}(t)\| &\leq c_4 e^{-\nu t} + \left\{ |b_1| \left[2 + \frac{c_1 c_2 (k^* - k_*)}{2k_*} \right] + \frac{c_2 |b_2 - 2ab_1| \sqrt{k_0}}{\sqrt{\gamma}} \right\} \|\mathbf{f}(t)\|_{\mathcal{L}_\infty} \end{aligned} \quad (21)$$

hold in the vector case as well, when $a \geq \sqrt{\gamma k_0}$.

III. REFERENCE MODEL MODIFICATION

Consider a multi-input multi-output (MIMO) uncertain linear system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (22)$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^q$ are the state and input of the system respectively, and $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times q}$ are unknown constant matrices satisfying the following matching conditions.

Assumption 3.1: Given a Hurwitz matrix $\mathbf{A}_m \in \mathbb{R}^{n \times n}$ and a matrix $\mathbf{B}_m \in \mathbb{R}^{n \times q}$ of full column rank, there exists a matrix $\mathbf{K}_1 \in \mathbb{R}^{n \times q}$ and a sign definite matrix $\Lambda \in \mathbb{R}^{q \times q}$ such that the following equations hold:

$$\begin{aligned} \mathbf{B} &= \mathbf{B}_m \Lambda \\ \mathbf{A} &= \mathbf{A}_m - \mathbf{B}\mathbf{K}_1^\top \end{aligned} \quad (23)$$

Remark 3.1: The sign definiteness of Λ corresponds to the conventional condition on the high-frequency gain matrix of MIMO systems (for example, see [15]). Without loss of generality, we assume that Λ is positive definite. The rest of the conditions for the existence of an adaptive controller are given by Eqs. (23).

As in the conventional model reference adaptive control (MRAC) framework, the objective is to design a control signal $\mathbf{u}(t)$ such that the the system (22) tracks a given reference model

$$\dot{\mathbf{x}}_r(t) = \mathbf{A}_m \mathbf{x}_r(t) + \mathbf{B}_m \mathbf{r}(t) \quad \mathbf{x}_r(0) = \mathbf{x}_0 \quad (24)$$

where $\mathbf{A}_m, \mathbf{B}_m$ are chosen according to performance specifications and satisfy Assumption 3.1 and $\mathbf{r}(t)$ is a bounded external command, which has a bounded derivative almost everywhere on $[0, \infty)$. Therefore, it is piecewise uniformly continuous on $[0, \infty)$ in the sense of the definition from [16]. We notice that discontinuous commands such as steps or square waves satisfy these conditions.

The system (22) can be written in the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}_m \mathbf{x}(t) + \mathbf{B}_m \mathbf{r}(t) + \mathbf{B}_m \Lambda [\mathbf{u}(t) - \Theta^\top \mathbf{f}(t)] \quad (25)$$

where $\Theta = [\mathbf{K}_1^\top \quad \mathbf{K}_2^\top]$, $\mathbf{f}(t) = [\mathbf{x}^\top(t) \quad \mathbf{r}^\top(t)]^\top$ and $\mathbf{K}_2^\top = \Lambda^{-1}$. Ideally, if the matrices \mathbf{K}_1 and \mathbf{K}_2 were known, one could set the control signal equal to

$$\varphi(t) = \Theta^\top \mathbf{f}(t) \quad (26)$$

thus reducing the system's dynamics to the given reference model. Although the reference model (24) can always be specified from the performance perspectives, the control signal (26), which is called an ideal control signal, cannot be implemented because the matrices \mathbf{K}_1 and \mathbf{K}_2 are unknown. Therefore, the adaptive version of it is implemented in the MRAC framework, that is,

$$\mathbf{u}(t) = \hat{\Theta}^\top(t) \mathbf{f}(t) \quad (27)$$

where $\hat{\Theta}(t)$ is the estimate of the ideal control gain Θ and is updated online according to the

adaptive law

$$\hat{\Theta}(t) = -\gamma \mathbf{f}(t) \mathbf{e}^\top(t) P B_m \quad \hat{\Theta}(0) = \Theta_0 \quad (28)$$

where $\gamma > 0$ is the adaptation rate, $\mathbf{e}(t) = \mathbf{x}(t) - \mathbf{x}_r(t)$ is the tracking error, and $P = P^\top > 0$ is the solution of the Lyapunov equation

$$A_m^\top P + P A_m = -Q \quad (29)$$

for some $Q = Q^\top > 0$. Introducing the parameter estimation error as $\tilde{\Theta}(t) = \hat{\Theta}(t) - \Theta$, the tracking error dynamics can be written in the form

$$\dot{\mathbf{e}}(t) = A_m \mathbf{e}(t) + B_m \Lambda \tilde{\Theta}^\top(t) \mathbf{f}(t) \quad (30)$$

Remark 3.2: We initialize the reference model at the system's initial condition, which is achievable in the state feedback framework. One can simply take a first sample of the state measurement as the initial condition for the reference model. However, for the sake of completeness, we will emphasize the discrepancies arising from the initialization error, where it is needed.

It is well known that this control architecture guarantees asymptotic tracking $\mathbf{x}(t) \rightarrow \mathbf{x}_r(t)$ as $t \rightarrow \infty$ while ensuring the boundedness of all closed-loop signals. However, the transient behavior of $\mathbf{x}(t)$ and $\mathbf{u}(t)$ cannot be guaranteed. In addition, high-frequency oscillations may be generated in the control signal for large adaptation rates.

To overcome this shortcoming of the MRAC design, we propose a control architecture that is based on the modification of the reference model (24). The rationale behind this modification is to drive the reference model toward the system during the transient phase, when the adaptive control signal $\mathbf{u}(t)$ is far from the ideal control signal $\varphi(t)$ because of the initial errors in the control gains. This motion of the reference model reduces the tracking error in the initial phase, thus allowing for a smoother adaptation even with high adaptation rate. Meanwhile, the modification term vanishes when the tracking is achieved. The proposed modified reference model has the form

$$\dot{\mathbf{x}}_m(t) = A_m \mathbf{x}_m(t) + B_m \mathbf{r}(t) + \lambda \mathbf{e}_m(t) \quad \mathbf{x}_m(0) = \mathbf{x}_0 \quad (31)$$

where $\lambda > 0$ is a design parameter and $\mathbf{e}_m(t) = \mathbf{x}(t) - \mathbf{x}_m(t)$ is the modified tracking error and the subscript m is introduced to distinguish between the original reference model (24) and the modified reference model (31), which is used for the actual control design. The original reference model and the ideal control signal will be used only for the comparison purposes. Similarly, we introduced a notation $\mathbf{e}_m(t)$ for the modified tracking error to distinguish from the tracking error $\mathbf{e}(t)$ between the system and the original reference model. With the proposed modification the modified error dynamics take the form

$$\dot{\mathbf{e}}_m(t) = (A_m - \lambda \mathbb{I}_n) \mathbf{e}_m(t) + B_m \Lambda [\mathbf{u}(t) - \Theta^\top \mathbf{f}(t)] \quad (32)$$

where \mathbb{I}_n denotes an n -dimensional identity matrix. The adaptive control $\mathbf{u}(t)$ is defined according to Eq. (27) as in MRAC design, but the adaptive law now is based on the modified tracking error $\mathbf{e}_m(t)$

$$\dot{\hat{\Theta}}(t) = -\gamma \mathbf{f}(t) \mathbf{e}_m^\top(t) P B_m \quad \hat{\Theta}(0) = \Theta_0 \quad (33)$$

With the application of the adaptive control $\mathbf{u}(t)$, the modified tracking error dynamics reduce to

$$\dot{\mathbf{e}}_m(t) = (A_m - \lambda \mathbb{I}_n) \mathbf{e}_m(t) + B_m \Lambda \tilde{\Theta}^\top(t) \mathbf{f}(t) \quad (34)$$

where the parameter estimation error $\tilde{\Theta}(t)$ is defined similarly to MRAC.

In the following analysis we will also need the control error that is defined as $\tilde{\mathbf{u}}(t) = \mathbf{u}(t) - \varphi(t)$. From the preceding constructions it follows that

$$\tilde{\mathbf{u}}(t) = \hat{\Theta}^\top(t) \mathbf{f}(t) - \Theta^\top \mathbf{f}(t) = \tilde{\Theta}^\top(t) \mathbf{f}(t) \quad (35)$$

Because the ideal control signal is the best achievable signal, we are interested in minimizing the control error $\tilde{\mathbf{u}}(t)$, as well as the state errors $\mathbf{e}(t)$ and $\mathbf{e}_m(t)$, both in transient and steady state by selecting proper values for the adaptation rate γ and feedback parameter λ . This is the main objective of the analysis in the following sections.

Remark 3.3: When $\lambda = 0$, the proposed control architecture is identical with the conventional MRAC, as the only difference from the MRAC design is the modification of the reference model by the term $\lambda \mathbf{e}_m(t)$. To emphasize this, we call the proposed control scheme a modified reference model MRAC or M-MRAC in short.

IV. ASYMPTOTIC PROPERTIES OF M-MRAC

The following theorem summarizes the asymptotic properties of the proposed M-MRAC architecture.

Theorem 4.1

Let the system (22) satisfy Assumption 3.1 and the reference model be given by Eq. (24). Let the system be controlled by the M-MRAC scheme given by Eqs. (27), (31), and (33). Then,

- 1) all closed-loop signals are bounded, and 2) $\mathbf{e}(t) \rightarrow 0$, $\mathbf{e}_m(t) \rightarrow 0$ and $\tilde{\mathbf{u}}(t) \rightarrow 0$ as $t \rightarrow \infty$.

Proof: Consider the following candidate Lyapunov function:

$$V(t) = \mathbf{e}_m^\top(t) P \mathbf{e}_m(t) + \frac{1}{\gamma} \text{trace} \left[\Lambda \tilde{\Theta}^\top(t) \tilde{\Theta}(t) \right] \quad (36)$$

Computing its derivative along the trajectories of the systems (33) and (34), we readily obtain

$$\dot{V}(t) = -\mathbf{e}_m^\top(t) Q \mathbf{e}_m(t) - 2\lambda \mathbf{e}_m^\top(t) P \mathbf{e}_m(t) \quad (37)$$

which implies that $\mathbf{e}_m(t)$, $\tilde{\Theta}(t) \in \mathcal{L}_\infty$, and $V(t) \in \mathcal{L}_\infty$. Because the reference model (31) can be viewed as an exponentially stable system with bounded inputs $\mathbf{r}(t)$ and $\mathbf{e}_m(t)$, it follows that $\mathbf{x}_m(t) \in \mathcal{L}_\infty$ as well. Hence, $\mathbf{x}(t) \in \mathcal{L}_\infty$. Because the inclusion $\tilde{\Theta}(t) \in \mathcal{L}_\infty$ implies that $\Theta(t) \in \mathcal{L}_\infty$, it follows that $\mathbf{u}(t) \in \mathcal{L}_\infty$. Then $\tilde{\mathbf{u}}(t) \in \mathcal{L}_\infty$, because $\varphi(t) \in \mathcal{L}_\infty$ by the definition

in Eq. (26). From the error dynamics (34) it follows that $\dot{\mathbf{e}}_m(t) \in \mathcal{L}_\infty$. The boundedness of the error signal $\mathbf{e}(t)$ can be obtained using the fact that $\mathbf{e}_m(t) - \mathbf{e}(t) = \mathbf{x}_r(t) - \mathbf{x}_m(t)$, and therefore

$$\frac{d}{dt} [\mathbf{e}_m(t) - \mathbf{e}(t)] = A_m [\mathbf{e}_m(t) - \mathbf{e}(t)] - \lambda \mathbf{e}_m(t) \quad (38)$$

Because $\mathbf{e}_m(t) \in \mathcal{L}_\infty$ and A_m is Hurwitz, it follows that $[\mathbf{e}_m(t) - \mathbf{e}(t)] \in \mathcal{L}_\infty$, and hence $\mathbf{e}(t) \in \mathcal{L}_\infty$. This completes the first part.

Integrating Eq. (37), we obtain

$$[\lambda_{\min}(Q) + 2\lambda\lambda_{\min}(P)] \int_0^t \|\mathbf{e}_m(\tau)\|^2 d\tau \leq V(0) - V(t) \quad (39)$$

where $\lambda_{\min}(Q)$ denotes the minimum eigenvalue of matrix Q and $\|\mathbf{e}_m\|$ indicates the 2-norm of the vector \mathbf{e}_m . Because $V(t) \in \mathcal{L}_\infty$, it follows that $\mathbf{e}_m(t) \in \mathcal{L}_2$. Also, from the error dynamics (34) it follows that $\dot{\mathbf{e}}_m(t) \in \mathcal{L}_\infty$. Therefore, $\mathbf{e}_m(t)$ is uniformly continuous. Application of Barbalat's lemma [17] (p. 19) results in $\mathbf{e}_m(t) \rightarrow 0$ as $t \rightarrow \infty$. Then, from Eq. (38) it follows that $\mathbf{e}_m(t) - \mathbf{e}(t) \rightarrow 0$, and hence $\mathbf{e}(t) \rightarrow 0$ as $t \rightarrow \infty$. Furthermore, from the dynamic equation (22), it follows that $\dot{\mathbf{x}}(t) \in \mathcal{L}_\infty$, because $\mathbf{x}(t) \in \mathcal{L}_\infty$ and $\mathbf{u}(t) \in \mathcal{L}_\infty$. Therefore, $\mathbf{x}(t)$ is uniformly continuous. It follows from the adaptive law (33) that $\dot{\tilde{\Theta}}(t) \in \mathcal{L}_\infty$; therefore, $\tilde{\Theta}(t)$ is uniformly continuous as well. Because $\mathbf{r}(t)$ is piecewise uniformly continuous by the assumption, the right-hand side of the error dynamics (34) is piecewise uniformly continuous; therefore, $\dot{\mathbf{e}}_m(t)$ is piecewise uniformly continuous. Because $\mathbf{e}_m(t)$ has a finite limit, it follows from the extended Barablat's lemma [16] that $\dot{\mathbf{e}}_m(t) \rightarrow 0$ as $t \rightarrow \infty$. Therefore, the relationship

$$\lim_{t \rightarrow \infty} \tilde{\Theta}^\top(t) \mathbf{f}(t) = 0 \quad (40)$$

holds. Then, it follows from Eq. (35) that $\tilde{\mathbf{u}}(t) \rightarrow 0$ as $t \rightarrow \infty$. The proof is complete. \square

Theorem 4.1 shows that M-MRAC architecture guarantees asymptotic tracking of not only the modified reference model (31), but also the reference model of the conventional MRAC architecture presented in (24). That is, the asymptotic performance of both the conventional MRAC and the proposed M-MRAC designs is equivalent. However, as it is shown in the next section the two designs' transient performance is quite different.

V. TRANSIENT PROPERTIES OF M-MRAC

In this section we show that the proper selection of the design parameters γ and λ results in controllable and quantifiable transient performance for both output and input signals of the M-MRAC architecture.

Theorem 5.1

Let conditions of Theorem 4.1 be satisfied. Then the tracking error can be decreased as desired in both \mathcal{L}_∞ and \mathcal{L}_2 senses by the proper choice of design parameters.

Proof: We recall that Theorem 4.1 ensures that the Lyapunov function $V(t)$ is nonincreasing. Because $\mathbf{e}(0) = 0$ by selection of the reference model, the following chain of relationships holds:

$$\lambda_{\min}(P) \|\mathbf{e}_m(t)\|^2 \leq V(t) \leq V(0) = \frac{1}{\gamma} \text{trace} \left[\Lambda \tilde{\Theta}^\top(0) \tilde{\Theta}(0) \right] \quad (41)$$

Therefore, we can write

$$\|\mathbf{e}_m(t)\| \leq \frac{\vartheta}{\sqrt{\lambda_{\min}(P)} \sqrt{\gamma}} \quad (42)$$

where the positive constant ϑ is defined as

$$\vartheta = \sqrt{\text{trace} \left[\Lambda \tilde{\Theta}^\top(0) \tilde{\Theta}(0) \right]} \quad (43)$$

Because the inequality (42) holds uniformly in t , we conclude that

$$\|\mathbf{e}_m(t)\|_{\mathcal{L}_\infty} \leq \frac{\beta_1}{\sqrt{\gamma}} \quad (44)$$

where $\beta_1 = \frac{\vartheta}{\sqrt{\lambda_{\min}(P)}}$. Similarly, the bound

$$\|\mathbf{e}_m(t)\|_{\mathcal{L}_2} \leq \frac{\beta_2}{\sqrt{\gamma}} \quad (45)$$

where $\beta_2 = \frac{\vartheta}{\sqrt{\lambda_{\min}(Q) + 2\lambda\lambda_{\min}(P)}}$ follows from Eq. (39).

To obtain the bounds on the actual error signal $\mathbf{e}(t)$, we notice from Eq. (38) that the inclusion $\mathbf{e}_m(t) \in \mathcal{L}_2$ implies the inclusion $\mathbf{e}_m(t) - \mathbf{e}(t) \in \mathcal{L}_2$, and hence $\mathbf{e}(t) \in \mathcal{L}_2$. Next, we recall from [13] (p. 199) that for the linear system (38) the following inequality holds:

$$\|\mathbf{e}(t) - \mathbf{e}_m(t)\|_{\mathcal{L}_p} \leq \lambda \|\Phi(t - \tau)\|_{\mathcal{L}_1} \|\mathbf{e}_m(t)\|_{\mathcal{L}_p} \quad (46)$$

for any $p \in [1, \infty]$, where $\Phi(t - \tau) = \exp[A_m(t - \tau)]$ is the system's state transition matrix. Because A_m is Hurwitz, there exist positive constants k_m such that $\|\Phi(t - \tau)\|_{\mathcal{L}_1} \leq k_m$ for all $\tau \geq 0$, $t \geq \tau$. The specific value of k_m is not of our interest because the matrix A_m is fixed from the performance perspectives, and the dependence of the transient bounds on the choice of A_m is not considered here. The norm bounds on the error signal $\mathbf{e}^0(t)$ readily follow from the inequalities (44–46)

$$\begin{aligned} \|\mathbf{e}(t)\|_{\mathcal{L}_\infty} &\leq \|\mathbf{e}_m(t)\|_{\mathcal{L}_\infty} + \lambda k_m \|\mathbf{e}_m(t)\|_{\mathcal{L}_\infty} \leq (1 + \lambda k_m) \frac{\beta_1}{\sqrt{\gamma}} \\ \|\mathbf{e}(t)\|_{\mathcal{L}_2} &\leq \|\mathbf{e}_m(t)\|_{\mathcal{L}_2} + \lambda k_m \|\mathbf{e}_m(t)\|_{\mathcal{L}_2} \leq (1 + \lambda k_m) \frac{\beta_2}{\sqrt{\gamma}} \end{aligned} \quad (47)$$

From the derived norm bounds in Eqs. (44), (45), and (47), it follows that the error signals $\mathbf{e}_m(t)$ and $\mathbf{e}(t)$ can be arbitrarily decreased by increasing the adaptation rate γ . However, while increasing λ improves the \mathcal{L}_2 performance and leaves intact the \mathcal{L}_∞ performance of the error signal $\mathbf{e}_m(t)$, it also increases the norm bounds on the error signal $\mathbf{e}(t)$. That is, λ cannot be selected arbitrarily and

should be related to γ in such a way that $\|\mathbf{e}(t)\|_{\mathcal{L}_\infty}$ and $\|\mathbf{e}(t)\|_{\mathcal{L}_2}$ are not increased when increasing the design parameters for the improved transient response. One way to achieve this goal is by setting

$$\lambda = c_0 \sqrt{\gamma} \quad (48)$$

where $c_0 > 0$ is to be selected from the control error perspective. In this case, the norm bounds on $\mathbf{e}(t)$ can be expressed as

$$\begin{aligned} \|\mathbf{e}(t)\|_{\mathcal{L}_\infty} &\leq \frac{\beta_1}{\sqrt{\gamma}} + k_m c_0 \beta_1 \\ \|\mathbf{e}(t)\|_{\mathcal{L}_2} &\leq \frac{\beta_2}{\sqrt{\gamma}} + k_m c_0 \beta_2 \end{aligned} \quad (49)$$

It follows from the preceding inequalities that the oscillations (\mathcal{L}_2 norm) in the tracking error $\mathbf{e}(t)$ are reduced by increasing λ and γ simultaneously according to Eq. (48). The overshoot (\mathcal{L}_∞ norm) is regulated by the choice of γ and matrix P [or Q in the Lyapunov equation (29)]. In the derived bounds, ϑ represents all unknown quantities [if we choose $\hat{\Theta}(0) = 0$, then $\vartheta = \sqrt{\text{trace}(\Lambda \Theta^\top \Theta)}$] and is independent of the design parameters λ and γ . \square

Remark 5.1: The derived bounds still hold, when the reference model cannot be initialized at the system's initial conditions. The difference is an additional term in ϑ , which is now given by the equation

$$\vartheta = \sqrt{\mathbf{e}_m^\top(0)P\mathbf{e}_m(0) + \text{trace}\left[\Lambda \tilde{\Theta}^\top(0)\tilde{\Theta}(0)\right]} \quad (50)$$

Next we analyze the transient performance of the control error signal $\tilde{\mathbf{u}}(t)$, which is defined by Eq. (35). Substituting $\tilde{\mathbf{u}}(t)$ into the tracking error dynamics (34), we can write

$$\dot{\mathbf{e}}_m(t) = (A_m - \lambda \mathbb{I}_n)\mathbf{e}_m(t) + B_m \Lambda \tilde{\mathbf{u}}(t) \quad (51)$$

We notice that $\tilde{\mathbf{u}}(t)$ does not explicitly depend on design parameters λ and γ . Instead, $\dot{\tilde{\mathbf{u}}}(t)$ depends on γ through the adaptive laws, and $\ddot{\tilde{\mathbf{u}}}(t)$ depends on λ through the tracking error dynamics. Differentiating $\tilde{\mathbf{u}}(t)$ and substituting the adaptive laws, we obtain

$$\dot{\tilde{\mathbf{u}}}(t) = -\gamma \rho(t) B_m^\top P \mathbf{e}_m(t) + \mathbf{h}(t) \quad (52)$$

where we denote $\rho(t) = \mathbf{f}^\top(t)\mathbf{f}(t)$ and $\mathbf{h}(t) = \tilde{\Theta}^\top(t)\dot{\mathbf{f}}(t)$. Differentiating Eq. (52) with respect to time, we obtain a second-order differential equation in $\tilde{\mathbf{u}}(t)$, which can be written in the following matrix form:

$$\begin{bmatrix} \dot{\mathbf{z}}_1(t) \\ \dot{\mathbf{z}}_2(t) \end{bmatrix} = \begin{bmatrix} 0_{q \times q} & I_{q \times q} \\ -\gamma \rho(t) G & -\lambda I_{q \times q} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1(t) \\ \mathbf{z}_2(t) \end{bmatrix} + \gamma \begin{bmatrix} 0_{q \times 1} \\ \mathbf{y}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{h}(t) \\ 0_{q \times 1} \end{bmatrix} \quad (53)$$

where notations $\mathbf{z}_1(t) = \tilde{\mathbf{u}}(t)$, $\mathbf{z}_2(t) = \dot{\tilde{\mathbf{u}}}(t) - \mathbf{h}(t)$, $G = B_m^\top P B_m \Lambda$, and $\mathbf{y}(t) = -\rho(t) B_m^\top \times P A_m \mathbf{e}_m(t) - \dot{\rho}(t) B_m^\top P \mathbf{e}_m(t)$ have been introduced. From the results of Sec. IV, it follows that $\rho(t)$

is bounded. That is, there exists a positive constant α_1 such that $\|\rho(t)\|_{\mathcal{L}_\infty} \leq \alpha_1$. Because $\dot{\mathbf{r}}(t)$ is bounded almost everywhere, it follows that the signals $\dot{\rho}(t)$ and $\mathbf{h}(t)$ are bounded almost everywhere on $[0, \infty)$. That is, there exist positive constants α_2, α_3 such that the inequalities $\|\dot{\rho}(t)\|_{\mathcal{L}_\infty} \leq \alpha_2$ and $\|\mathbf{h}(t)\|_{\mathcal{L}_\infty} \leq \alpha_3$ hold almost everywhere on $[0, \infty)$. Therefore, Eq. (53) can be considered as a second-order linear equation with time-varying coefficients in $\tilde{\mathbf{u}}(t)$ in the Caratheodory sense [18] (Theorem 2.1.1). Although Eq. (53) is nonautonomous, it can be still inferred that the adaptation rate γ determines the frequency of $\tilde{\mathbf{u}}(t)$ and hence the frequency of the control signal $\mathbf{u}(t)$, because the ideal control $\varphi(t)$ is in the low-frequency range. Therefore, increasing γ increases the oscillations in $\mathbf{u}(t)$ as is the case for the conventional MRAC design. On the other hand, λ determines the damping ratio. Therefore, increasing λ suppresses the oscillations in $\tilde{\mathbf{u}}(t)$ and hence in the control signal $\mathbf{u}(t)$. That is, by selecting a proper value for λ the desired performance can be achieved. This is the main difference from the MRAC design, which results when $\lambda = 0$.

A proper value for the parameter λ can be selected from the perspective of minimizing the norm bound for $\tilde{\mathbf{u}}(t)$. To this end, we decouple Eq. (53) by noting that the matrix G is symmetric and positive definite. Therefore, there exists an orthogonal matrix T such that $S = TGT^T$ is diagonal with positive entries s_i , $i = 1, \dots, q$. That is, introducing the new variables $\mathbf{v}_1 = T\mathbf{z}_1$ and $\mathbf{v}_2 = T\mathbf{z}_2$, Eq. (53) can be written component-wise

$$\begin{bmatrix} \dot{v}_{1i}(t) \\ \dot{v}_{2i}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\gamma s_i \rho(t) & -\lambda \end{bmatrix} \begin{bmatrix} v_{1i}(t) \\ v_{2i}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \gamma \end{bmatrix} \bar{y}_i(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \bar{h}_i(t) \quad (54)$$

for each $i = 1, \dots, q$, where we denote $\bar{\mathbf{y}} = T\mathbf{y}$ and $\bar{\mathbf{h}} = T\mathbf{h}$. We notice that T preserves the vector norms because it is orthogonal. That is, $\|\bar{\mathbf{v}}_1\| = \|\mathbf{z}_1\| = \|\tilde{\mathbf{u}}\|$, $\|\bar{\mathbf{y}}\| = \|\mathbf{y}\|$, and $\|\bar{\mathbf{h}}\| = \|\mathbf{h}\|$.

The selection of λ and the bound on $\tilde{\mathbf{u}}(t)$ is given by the following theorem.

Theorem 5.2

Let conditions of Theorem 4.1 be satisfied. Then the control error $\tilde{\mathbf{u}}(t)$ can be bounded by the sum of an exponentially decaying term and a term that can be decreased as desired by increasing the adaptation rate γ , when the feedback gain λ is proportional to $\sqrt{\gamma}$.

Proof: We consider the following cases.

Case 1: $\rho(t) \neq 0$ for all $t \in [0, \infty)$ [for example, when $\mathbf{r}(t) \neq 0$]. In this case there exists a positive constant ρ_* such that $\rho(t) \geq \rho_*$ on $[0, \infty)$.

Case 2: $\rho(t_i^*) = 0$, $t_i^* \in [0, \infty)$, $i = 1, 2, \dots$ (for example, tracking a sinusoidal command). In this case we can select a positive constant β_0 and intervals $t_i^l \leq t_i^* \leq t_i^u$ with $t_{i+1}^l \geq t_i^u$ such that $\rho(t) \leq \beta_0$ on $[t_i^l, t_i^u]$, and $\rho(t) \geq \beta_0$ on $[t_{i-1}^u, t_i^l]$ and $[t_i^u, t_{i+1}^l]$.

Case 3: $\rho(t) = 0$ on some interval $[t_1, t_2]$ (trivial case). In this case $\mathbf{r}(t) = 0$ and $\mathbf{x}(t) = 0$ on $[t_1, t_2]$. Therefore, $\mathbf{u}(t) = 0$, $\mathbf{u}^0(t) = 0$, and $\tilde{\mathbf{u}}(t) = 0$ on $[t_1, t_2]$.

Now consider Case 1. In this case, Eq. (54) is in the form of Eq. (3) with $k(t) = s_i \rho(t)$, $\lambda = 2a$ and two external inputs

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \gamma \bar{y}_i(t) \quad \text{and} \quad \begin{bmatrix} 1 \\ 0 \end{bmatrix} \bar{h}_i(t)$$

If we denote $k_0 = (\alpha_1 + \rho_*)/2$, $\omega_i^2 = \gamma s_i k_0$, and select $\lambda = 2\sqrt{\omega_i} = \sqrt{2\gamma s_i(\alpha_1 + \rho_*)}$ for each channel, the inequalities (18) can be applied. Extending Eq. (18) to the two input case with $B = [0 \ \gamma]^\top$, $f(t) = \bar{y}_i(t)$ for the first input and $B = [1 \ 0]^\top$, $f(t) = \bar{h}_i(t)$ for the second input, we obtain

$$|v_{1i}(t)| \leq c_3 e^{-\nu t} \sqrt{v_{1i}^2(0) + v_{2i}^2(0)} + \frac{1}{s_i \rho_*} \|\bar{y}_i(t)\|_{\mathcal{L}_\infty} + \frac{c_1 \sqrt{k_0}}{s_i \rho_* \sqrt{\gamma}} \|\bar{h}_i(t)\|_{\mathcal{L}_\infty} \quad (55)$$

for each $i = 1, \dots, q$. This inequality also holds if $\lambda > 2\sqrt{\omega_i} = \sqrt{\gamma s_i k_0}$. Therefore, selecting

$$\lambda = c_0 \sqrt{\gamma}, \quad (56)$$

where $c_0 = \sqrt{2s^*(\alpha_1 + \rho_*)}$ and $s^* = \max\{s_i, i = 1, \dots, q\}$ results in the inequalities

$$|v_{1i}(t)| \leq c_3 e^{-\nu t} \sqrt{v_{1i}^2(0) + v_{2i}^2(0)} + \frac{1}{s_* \rho_*} \|\bar{y}_i(t)\|_{\mathcal{L}_\infty} + \frac{c_1 \sqrt{k_0}}{s_* \rho_* \sqrt{\gamma}} \|\bar{h}_i(t)\|_{\mathcal{L}_\infty} \quad (57)$$

for all channels simultaneously, where $s_* = \min\{s_i, i = 1, \dots, q\}$. The similar inequality holds for the vector $\mathbf{v}_1(t)$. That is,

$$\|\mathbf{v}_1(t)\| \leq c_4 e^{-\nu t} + \frac{1}{s_* \rho_*} \|\bar{\mathbf{y}}(t)\|_{\mathcal{L}_\infty} + \frac{c_1 \sqrt{k_0}}{s_* \rho_* \sqrt{\gamma}} \|\bar{\mathbf{h}}(t)\|_{\mathcal{L}_\infty} \quad (58)$$

where $c_4 = \sqrt{\|\mathbf{v}_1(0)\|^2 + \|\mathbf{v}_2(0)\|^2}$.

We notice that the functions $\mathbf{y}(t)$ and $\mathbf{h}(t)$ may be discontinuous at some points, which are at most countable in the interval $[0, \infty)$. However, the inequities (55) have been derived from the corresponding integral equations, in evaluation of which these points of discontinuities can be ignored. Therefore, in evaluating the \mathcal{L}_∞ norm of $\mathbf{v}_1(t)$ and $\mathbf{v}_2(t)$, the inequalities $\|\dot{\rho}(t)\|_{\mathcal{L}_\infty} \leq \alpha_2$ and $\|\mathbf{h}(t)\|_{\mathcal{L}_\infty} \leq \alpha_3$ can be extended to the whole interval $[0, \infty)$. This results in the inequality

$$\|\bar{\mathbf{y}}(t)\|_{\mathcal{L}_\infty} \leq \|\mathbf{y}(t)\|_{\mathcal{L}_\infty} \leq (\alpha_1 \|B_m^\top P A_m\| + \alpha_2 \|B_m^\top P\|) \|\mathbf{e}_m(t)\|_{\mathcal{L}_\infty} \quad (59)$$

Substituting the bound on the error $\mathbf{e}_m(t)$ from Eq. (44) and taking into account that $\|\bar{\mathbf{u}}(t)\| = \|\mathbf{v}_1(t)\|$, we obtain

$$\|\bar{\mathbf{u}}(t)\| \leq c_4 e^{-\nu t} + \frac{c_5}{s_* \rho_* \sqrt{\gamma}} \quad (60)$$

where ν is proportional to $\sqrt{\gamma}$ and the constant

$$c_5 = (\alpha_1 \|B_m^\top P A_m\| + \alpha_2 \|B_m^\top P\|) \beta_1 + \alpha_3 c_1 \sqrt{k_0}$$

is independent of γ . That is, in Case 1, $\tilde{\mathbf{u}}(t)$ is bounded by the sum of an exponentially decaying term and a term that decreases in γ .

Next, we consider Case 2. For definiteness let $t_1^l = 0$. Because $\rho(t) \leq \beta_0$ on the interval $[t_1^l, t_1^u]$, we can write

$$\|\tilde{\mathbf{u}}(t)\| \leq \sqrt{\mathbf{f}^\top(t)\mathbf{f}(t)} \sqrt{\text{trace}[\tilde{\Theta}^\top(t)\tilde{\Theta}(t)]} \leq \sqrt{\beta_0} \sqrt{\text{trace}[\tilde{\Theta}^\top(t)\tilde{\Theta}(t)]} \quad (61)$$

From the inequality (41) it follows that

$$\sqrt{\text{trace}[\tilde{\Theta}^\top(t)\tilde{\Theta}(t)]} \leq \frac{\vartheta}{\lambda_{\min}(\Lambda)} \quad (62)$$

Therefore, we obtain the inequality

$$\|\tilde{\mathbf{u}}(t)\| \leq \frac{\vartheta}{\lambda_{\min}(\Lambda)} \sqrt{\beta_0} \quad (63)$$

for all $t \in [t_1^l, t_1^u]$. If we select

$$\beta_0 = \gamma^{-\frac{1}{3}} \quad (64)$$

the inequality (63) results in

$$\|\tilde{\mathbf{u}}(t)\| \leq \frac{\vartheta}{\lambda_{\min}(\Lambda)} \gamma^{-\frac{1}{6}} \quad (65)$$

Further, from the error dynamics of Eq. (51), it follows that

$$\|\mathbf{e}_m(t)\| \leq \|B_m\| \frac{\vartheta \lambda_{\max}(\Lambda)}{\lambda_{\min}(\Lambda)} \sqrt{\beta_0} \int_0^t \|\exp[(A_m - \lambda \mathbb{I}_n)(t - \tau)]\| d\tau \quad (66)$$

Because A_m is Hurwitz, it follows that $\|\exp[(A_m - \lambda \mathbb{I}_n)t]\| \leq \exp(-\lambda t)$. Therefore,

$$\|\mathbf{e}_m(t)\| \leq \|B_m\| \frac{\vartheta \lambda_{\max}(\Lambda)}{\lambda_{\min}(\Lambda)} \frac{\sqrt{\beta_0}}{\lambda} \quad (67)$$

on the interval $[t_1^l, t_1^u]$. With the selection of λ according to Eq. (56) and β_0 according to Eq. (64), the inequality (67), results in

$$\|\mathbf{e}_m(t)\| \leq \|B_m\| \frac{\vartheta \lambda_{\max}(\Lambda)}{\lambda_{\min}(\Lambda)} c_0 \gamma^{-\frac{2}{3}} \quad (68)$$

Then, from Eq. (52) the following bound can be derived:

$$\|\dot{\tilde{\mathbf{u}}}(t) - \mathbf{h}(t)\| \leq c_0 \|B_m^\top P\|_2 \|B_m\|_2 \frac{\partial \lambda_{\max}(\Lambda)}{\lambda_{\min}(\Lambda)} \equiv c_6 \quad (69)$$

which is independent of γ . The inequalities (65) and (69) hold on the interval $[t_1^l, t_1^u]$; therefore,

$$\begin{aligned} \|\tilde{\mathbf{u}}(t_1^u)\| &\leq \frac{\vartheta}{\lambda_{\min}(\Lambda)} \gamma^{-\frac{1}{6}} \\ \|\dot{\tilde{\mathbf{u}}}(t_1^u) - \mathbf{h}(t_1^u)\| &\leq c_6 \end{aligned} \quad (70)$$

On the next interval $[t_1^u, t_2^l]$, we have $\rho(t) > \beta_0$; therefore, the inequality (60) can be applied with $\tilde{\mathbf{u}}(0)$, $\tilde{\mathbf{u}}(0)$ and ρ_* replace with $\tilde{\mathbf{u}}(t_1^u)$, $\dot{\tilde{\mathbf{u}}}(t_1^u) - \mathbf{h}(t_1^u)$ and β_0 , respectively. This results in the inequality

$$\|\tilde{\mathbf{u}}(t)\| \leq c_5 \gamma^{-\frac{1}{6}} + c_3 \exp[-\nu(t - t_1^u)] \sqrt{\frac{\vartheta^2}{\lambda_{\min}^2(\Lambda)} \gamma^{-\frac{2}{6}} + c_6^2} \quad (71)$$

for all $t \in [t_1^u, t_2^l]$.

Applying the inequalities (60) and (71) in sequence, we obtain a bound on $\tilde{\mathbf{u}}(t)$ on the interval $[0, \infty)$ as the sum of an exponentially decaying term with the rate of decay ν and a term that can be decreased as desired by increasing the adaptation rate. This bound can be written as

$$\|\tilde{\mathbf{u}}(t)\| \leq \beta_3 e^{-\nu t} + \beta_4 \gamma^{-\frac{1}{6}} \quad (72)$$

where β_3 and β_4 are obtained from Eqs. (60) and (71). The proof is complete. \square

Remark 5.2: Using the inequality (72), it can be shown that the tracking error can be bounded by the sum of an exponentially decaying term and a term that can be decreased as desired by increasing the adaptation rate γ . Because A_m is Hurwitz, it follows that there exist positive constants b_0 and ν_0 such that $\|e^{A_m t}\| \leq b_0 e^{-\nu_0 t}$. Then, it follows from the error dynamics (30) that

$$\begin{aligned} \|\mathbf{e}(t)\| &\leq b_0 \|\mathbf{e}(0)\| e^{-\nu_0 t} + b_0 \|B\|_2 \int_0^t \exp[-\nu_0(t - \tau)] [\beta_3 e^{-\nu \tau} + \beta_4 \gamma^{-\frac{1}{6}}] d\tau \\ &\leq b_0 \|\mathbf{e}(0)\| e^{-\nu_0 t} + b_0 \|B\|_2 \left[\frac{\beta_3}{\nu - \nu_0} (e^{-\nu_0 t} - e^{-\nu t}) + \frac{\beta_4}{\nu} (1 - e^{-\nu t}) \gamma^{-\frac{1}{6}} \right] \end{aligned}$$

Recalling that ν is proportional to $\sqrt{\gamma}$, the inequality (73) can be expressed in the form

$$\|\mathbf{e}(t)\| \leq \beta_5 e^{-\nu^* t} + \beta_6 \gamma^{-\frac{1}{3}} \quad (73)$$

where $\nu^* = \min[\nu, \nu_0]$.

Remark 5.3: We notice that there are some uncertainties in setting λ according to the relationship (56) because of the matrix Λ and α_1 and β_0 . Usually, Λ represents the loss of control effectiveness; hence,

$\|\Lambda\| < 1$. Therefore, λ_{\max} can be estimated by 1, and s^* can be set to $s^* = \lambda_{\max}(B_m^\top P B_m)$. For the parameter $(\alpha_1 + \beta_0)/2$ a good estimate is $\rho_0 = \frac{1}{2}[\|\mathbf{x}_r(t)\|_{\mathcal{L}_\infty}^2 + \|\mathbf{r}(t)\|_{\mathcal{L}_\infty}^2]$. Alternatively, $\rho(t)$ can be used instead of $(\alpha_1 + \beta_0)/2$, thus making λ time variant. In this case, Theorems 4.1 and 5.1 still remain valid. In simulations, this selection gives a good performance for the control signal as well.

VI. DISTURBANCE REJECTION

In this section we analyze the disturbance rejection properties of the M-MRAC architecture. To this end, we insert a bounded but otherwise unknown disturbance signal $\mathbf{d}(t)$ into the system (22)

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}[\mathbf{u}(t) - \mathbf{d}(t)] \quad (74)$$

In this case, the ideal control signal has the form

$$\boldsymbol{\varphi}(t) = \boldsymbol{\Theta}^\top \mathbf{f}(t) + \mathbf{d}(t) \quad (75)$$

The corresponding adaptive control is given by

$$\mathbf{u}(t) = \hat{\boldsymbol{\Theta}}^\top(t) \mathbf{f}(t) + \hat{\mathbf{d}}(t) \quad (76)$$

where $\hat{\mathbf{d}}(t)$ is the estimate of a constant vector $\bar{\mathbf{d}}$, which is the constant part (or an average value) of $\mathbf{d}(t)$. It is well known that to prevent parameter drift it is necessary to use a robust adaptive law. Because we assume no prior knowledge of the parameter bounds, we use σ -modification type (for example, see [19] for details) adaptive laws to generate the estimates $\hat{\boldsymbol{\Theta}}(t)$ and $\hat{\mathbf{d}}(t)$:

$$\begin{aligned} \dot{\hat{\boldsymbol{\Theta}}}(t) &= -\gamma \mathbf{f}(t) \mathbf{e}_m^\top(t) P B_m - \sigma \hat{\boldsymbol{\Theta}}(t) \\ \dot{\hat{\mathbf{d}}}(t) &= -\gamma B_m^\top P \mathbf{e}_m(t) - \sigma \hat{\mathbf{d}}(t) \end{aligned} \quad (77)$$

The rest of the control structure is the same as in the disturbance-free case, including the input-output error relationship (51).

Here, we can show only ultimate boundedness of all closed-loop signals by means of the candidate Lyapunov function

$$V(t) = \mathbf{e}_m^\top(t) P \mathbf{e}_m(t) + \frac{1}{\gamma} \text{trace} \left[\Lambda \tilde{\boldsymbol{\Theta}}^\top(t) \tilde{\boldsymbol{\Theta}}(t) + \Lambda \tilde{\mathbf{d}}^\top(t) \tilde{\mathbf{d}}(t) \right] \quad (78)$$

where $\tilde{\mathbf{d}}(t) = \hat{\mathbf{d}}(t) - \bar{\mathbf{d}}$. Its derivative computed along the trajectories of the systems (34) and (77) takes the form

$$\begin{aligned} \dot{V}(t) &= -\mathbf{e}_m^\top(t) Q \mathbf{e}_m(t) - 2\lambda \mathbf{e}_m^\top(t) P \mathbf{e}_m(t) + 2\mathbf{e}_m^\top(t) P B_m \Lambda [\bar{\mathbf{d}} - \mathbf{d}(t)] \\ &\quad - 2\frac{\sigma}{\gamma} \text{tr} \left[\Lambda \tilde{\boldsymbol{\Theta}}^\top(t) \hat{\boldsymbol{\Theta}}(t) + \Lambda \tilde{\mathbf{d}}(t) \hat{\mathbf{d}}^\top(t) \right] \end{aligned} \quad (79)$$

Completing the squares in Eq. (79), we obtain

$$\dot{V}(t) \leq -a_1 \|\mathbf{e}_m(t)\|^2 - \frac{\sigma}{\gamma} \lambda_{\min}(\Lambda) \left[\|\tilde{\Theta}(t)\|_F^2 + \|\tilde{\mathbf{d}}(t)\|^2 \right] + c \quad (80)$$

where $a_1 = \lambda_{\min}(Q) + 2\lambda_{\min}(P)\lambda - 1$, $c = d_*^2 + \frac{\sigma}{\gamma} \lambda_{\max}(\Lambda) (\theta^2 + d_0^2)$, $d_* = \|PB_m \times \Lambda[\tilde{\mathbf{d}} - \mathbf{d}(t)]\|_{\mathcal{L}_\infty}$, $\theta = \|\Theta\|_F$, the subscript F denotes the Frobenius matrix norm, and $d_0 = \|\tilde{\mathbf{d}}\|$. Choosing Q such that $\lambda_{\min}(Q) - 1 \geq 0$, we conclude that $\dot{V}(t)$ is negative semi-definite outside the compact set

$$\Omega = \left\{ (\mathbf{e}_m, \tilde{\Theta}, \tilde{\mathbf{d}}) : a_1 \|\mathbf{e}_m\|^2 + \frac{\sigma}{\gamma} \lambda_{\min}(\Lambda) \left(\|\tilde{\Theta}\|_F^2 + \|\tilde{\mathbf{d}}\|^2 \right) \leq c \right\}$$

Therefore, the signals $\mathbf{e}_m(t)$, $\tilde{\Theta}(t)$, $\tilde{\mathbf{d}}(t)$ are uniformly ultimately bounded. The boundedness of $\mathbf{e}(t)$ follows from the inequality (46). Also, the boundedness of error signals implies the boundedness of $\mathbf{x}(t)$, $\hat{\Theta}(t)$, and $\hat{\mathbf{d}}(t)$. Hence, $\mathbf{u}(t)$ and $\tilde{\mathbf{u}}(t)$ are bounded as well.

From the boundedness of the system, it follows that there exist constants $k_1^* > 0$, $k_2^* > 0$ such that

$$\text{tr} \left[\Lambda \tilde{\Theta}^\top(t) \hat{\Theta}(t) + \tilde{\mathbf{d}}^\top(t) \hat{\mathbf{d}}(t) \Lambda \right] \leq k_1^* \quad (81)$$

$$\text{tr} \left[\Lambda \tilde{\Theta}^\top(t) \tilde{\Theta}(t) + \tilde{\mathbf{d}}^\top(t) \tilde{\mathbf{d}}(t) \Lambda \right] \leq k_2^* \quad (82)$$

Then, it follows from Eq. (37) that $\dot{V}(t) \leq 0$ if

$$\|\mathbf{e}_m(t)\| \geq \frac{d_*}{a_2} + \sqrt{\left(\frac{d_*}{a_2}\right)^2 + \frac{2\sigma}{a_2\gamma} k_1^*} \quad (83)$$

where we denote $a_2 = \lambda_{\min}(Q) + 2\lambda_{\min}(P)\lambda$. To obtain a bound on $\|\mathbf{e}_m(t)\|$, we consider a Lyapunov level set defined as

$$L = \{(\mathbf{e}_m, \tilde{K}_1, \tilde{K}_2, \tilde{\mathbf{d}}) : V(\mathbf{e}_m, \tilde{K}_1, \tilde{K}_2, \tilde{\mathbf{d}}) = V^*\}$$

with

$$V^* \triangleq \lambda_{\max}(P) \left(\frac{d_*}{a_2} + \sqrt{\left(\frac{d_*}{a_2}\right)^2 + \frac{2\sigma}{a_2\gamma} k_1^*} \right)^2 + \frac{1}{\gamma} k_2^* \quad (84)$$

It is easy to see that $\dot{V}(t) \leq 0$ whenever $V(t) \geq V^*$. Indeed, from the definition of $V(t)$ it follows that

$$\lambda_{\max}(P) \|\mathbf{e}_m(t)\|^2 + \frac{1}{\gamma} k_2^* \geq V(t) \geq V^* \quad (85)$$

which implies that $\|\mathbf{e}_m(t)\|$ satisfies the inequality (83). Therefore, $V(t) \leq V^*$ for all t . On the other hand, from the definition of $V(t)$, we have

$$\lambda_{\min}(P) \|\mathbf{e}_m(t)\|^2 \leq \mathbf{e}_m^\top(t) P \mathbf{e}_m(t) \leq V(t) \quad (86)$$

Therefore,

$$\|\mathbf{e}_m(t)\| \leq \sqrt{\frac{\lambda_{\max}(P)}{\lambda_{\min}(P)} \left[\frac{d_*}{a_2} + \sqrt{\left[\frac{d_*}{a_2} \right]^2 + \frac{2\sigma}{a_2 \gamma} k_1^*} \right]^2 + \frac{1}{\gamma \lambda_{\min}(P)} k_2^*} \quad (87)$$

Because the inequality (87) holds uniformly in t , $\|\mathbf{e}_m(t)\|_{\mathcal{L}_\infty}$ satisfies the same bound. The bound on the actual tracking error $\|\mathbf{e}(t)\|_{\mathcal{L}_\infty}$ follows from the inequality (46) and has the form

$$\|\mathbf{e}(t)\|_{\mathcal{L}_\infty} \leq (1 + \lambda k_m) \sqrt{\frac{\lambda_{\max}(P)}{\lambda_{\min}(P)} \left[\frac{d_*}{a_2} + \sqrt{\left[\frac{d_*}{a_2} \right]^2 + \frac{2\sigma}{a_2 \gamma} k_1^*} \right]^2 + \frac{1}{\gamma \lambda_{\min}(P)} k_2^*} \quad (88)$$

We notice that the derived bound on $\|\mathbf{e}(t)\|_{\mathcal{L}_\infty}$ cannot be arbitrarily decreased by increasing the design parameters λ and γ . With $\lambda = c_0 \sqrt{\gamma}$, the following asymptotic bound can be written:

$$\lim_{\gamma \rightarrow \infty} \|\mathbf{e}(t)\|_{\mathcal{L}_\infty} \leq k_m \sqrt{\frac{\lambda_{\max}(P)}{\lambda_{\min}^3(P)} d_*^2 + \frac{c_0}{\lambda_{\min}(P)} k_2^*} \quad (89)$$

which still can be reduced by increasing $\lambda_{\min}(P)$.

The bound on $\|\dot{\mathbf{u}}(t)\|_{\mathcal{L}_\infty}$ is derived as follows. Differentiating $\dot{\mathbf{u}}(t)$ and substituting the adaptive laws, we obtain

$$\dot{\mathbf{u}}(t) = -\gamma \rho(t) B_m^\top P \mathbf{e}_m(t) - \sigma[\varphi(t) + \dot{\mathbf{u}}(t)] + \mathbf{h}(t) \quad (90)$$

where now $\rho(t) = \mathbf{f}^\top(t) \mathbf{f}(t) + 1$. Differentiating Eq. (90) with respect to time, we obtain the following second-order differential equation:

$$\begin{aligned} \ddot{\mathbf{u}}(t) + (\lambda + \sigma) \dot{\mathbf{u}}(t) + [\lambda \sigma \mathbb{I}_q + \gamma \rho(t) G] \dot{\mathbf{u}}(t) &= -\gamma \rho(t) B_m^\top P A_m \mathbf{e}_m(t) \\ &\quad - \gamma \dot{\rho}(t) B_m^\top P \mathbf{e}_m(t) + \lambda \mathbf{h}(t) + \dot{\mathbf{h}}(t) - \sigma[\dot{\varphi}(t) + \lambda \varphi(t)] \end{aligned} \quad (91)$$

Again using the transformations $\mathbf{v}_1(t) = T \mathbf{z}_1(t) = T \dot{\mathbf{u}}(t)$ and $\mathbf{v}_2(t) = T \mathbf{z}_2(t) = T[\dot{\mathbf{u}} - \mathbf{h}(t) + \sigma \varphi(t)]$ with $TGT^\top = S$, we can write the component-wise

equations as

$$\begin{aligned} \begin{bmatrix} \dot{v}_{1i}(t) \\ \dot{v}_{2i}(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -\lambda\sigma - \gamma s_i \rho(t) & -\lambda - \sigma \end{bmatrix} \begin{bmatrix} v_{1i}(t) \\ v_{2i}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \gamma \end{bmatrix} \bar{y}_i(t) \\ &+ \begin{bmatrix} 1 \\ 0 \end{bmatrix} [\bar{h}_i(t) - \sigma \varphi_i(t)] \end{aligned} \quad (92)$$

for each $i = 1, \dots, q$. Equation (92) is in the form of Eq. (3) with $k(t) = (\lambda\sigma/\gamma) + s_i\rho(t)$ and $\lambda + \sigma = 2a$. Because in this case $k(t) \geq s_* + (\lambda\sigma/\gamma) = k_* > 0$, analogous to the preceding case, the following upper bound is derived:

$$\|\tilde{\mathbf{u}}(t)\| \leq c_4 e^{-\nu t} + \frac{c_5}{\rho_* s_* \sqrt{\gamma}} + \sigma c_7 \|\varphi(t)\|_{\mathcal{L}_\infty} \quad (93)$$

where $c_7 = \alpha_3 c_1 \sqrt{k_0}$. Clearly, in this case also the bound on the input error decreases with the increase of the adaptation rate. The difference is the additional term $\sigma c_7 \|\varphi(t)\|_{\mathcal{L}_\infty}$ in the input error bound, which is proportional to σ , usually chosen to be small.

VII. SCALAR EXAMPLE

In this section we demonstrate the improved performance of the M-MRAC over the MRAC in simulations for a scalar system given by

$$\dot{x}(t) = ax(t) + bu(t) \quad x(0) = x_0 \quad (94)$$

with $a = 3$ and $b = 1$. The parameters of the reference model

$$\dot{x}_r(t) = -a_m x_r(t) + b_m r(t) \quad x_r(0) = x_r \quad (95)$$

are set $a_m = 2$ and $b_m = 2$. The external command is chosen to be 1) a step function at time $t = 3$ s with the magnitude r_0 , 2) a sinusoid of frequency 2 rad/s and of amplitude r_0 , and 3) a square wave of frequency 2 rad/s and of amplitude r_0 . Here λ is set to $2\sqrt{\gamma(x_0^2 + r_0^2)}$. The response to the step command of MRAC and M-MRAC with $\gamma = 100$, $r_0 = 1$, and $x_0 = 0$ is displayed in Fig. 2. The improvement is obvious even with a small adaptation rate. No oscillations are observed in the M-MRAC response contrary to the conventional MRAC response. Next, we increase the adaptation rate to $\gamma = 1000$. The corresponding performances of MRAC and M-MRAC are presented in Fig. 3. It can be seen that the output tracking performance of MRAC gets better at the expense of high-frequency control signal, while the output and input tracking performances of M-MRAC are both substantially improved. Further increase of γ for the M-MRAC design results in the performance indistinguishable from that of the reference model both for input and output signals as can be observed from Fig. 4.

To show the uniform performance of the M-MRAC in external input signals, we run simulations for sinusoidal and square wave inputs with $\gamma = 1000$. The corresponding results are displayed in Figs. 5 and 6. It can be observed that

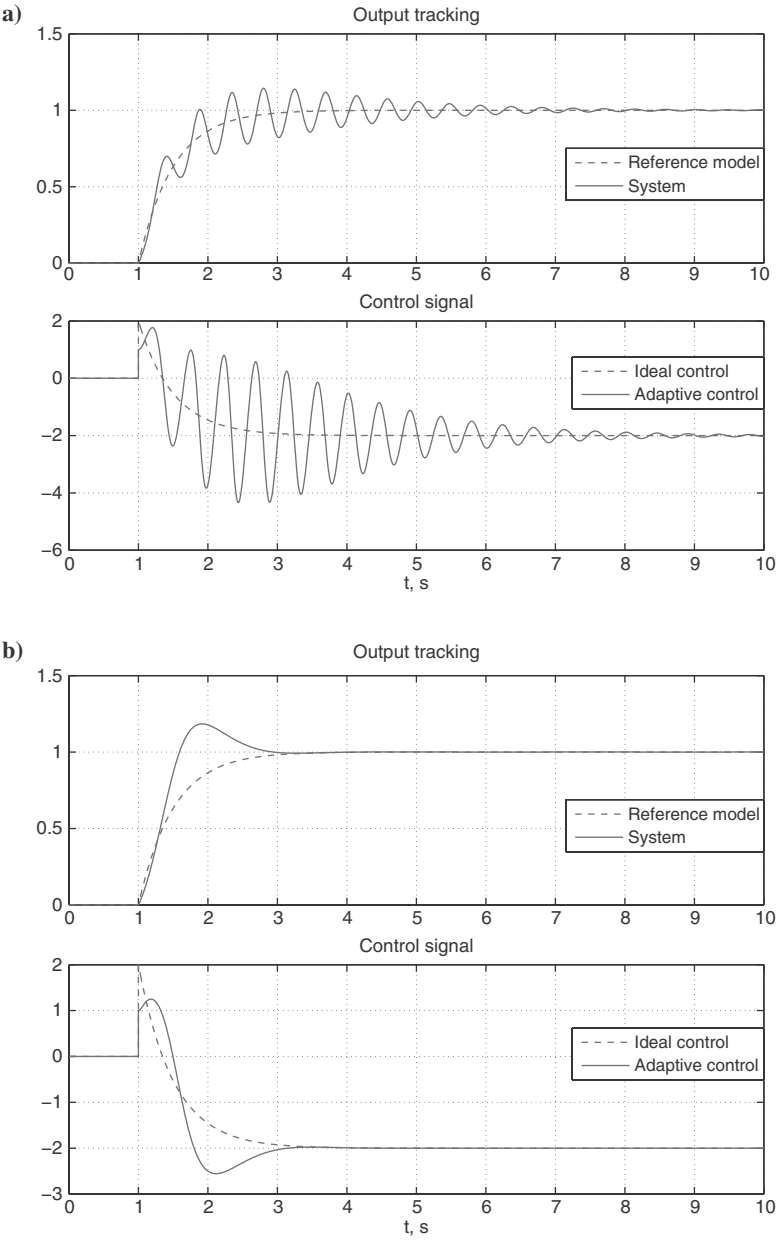


Fig. 2 Step response with $\gamma = 100$: a) MRAC performance and b) M-MRAC performance.

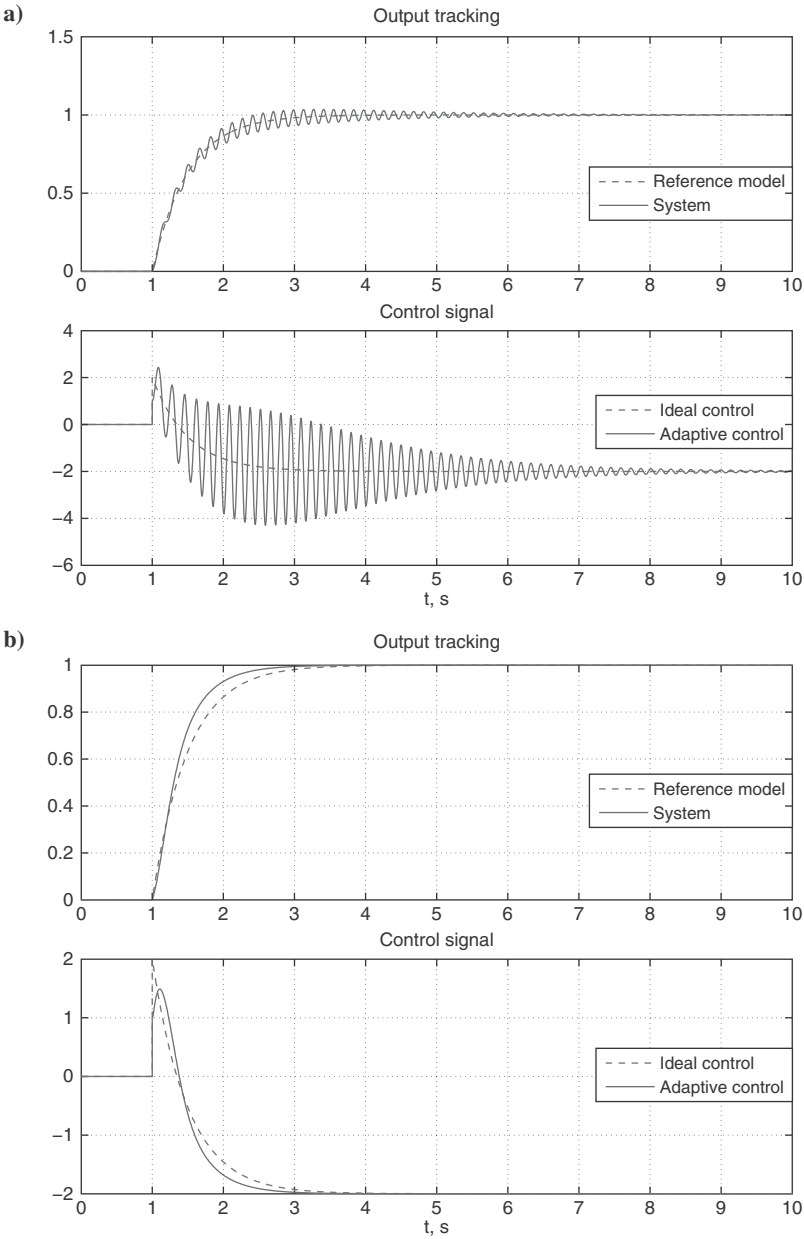


Fig. 3 Step response with $\gamma = 1000$: a) MRAC performance and b) M-MRAC performance.

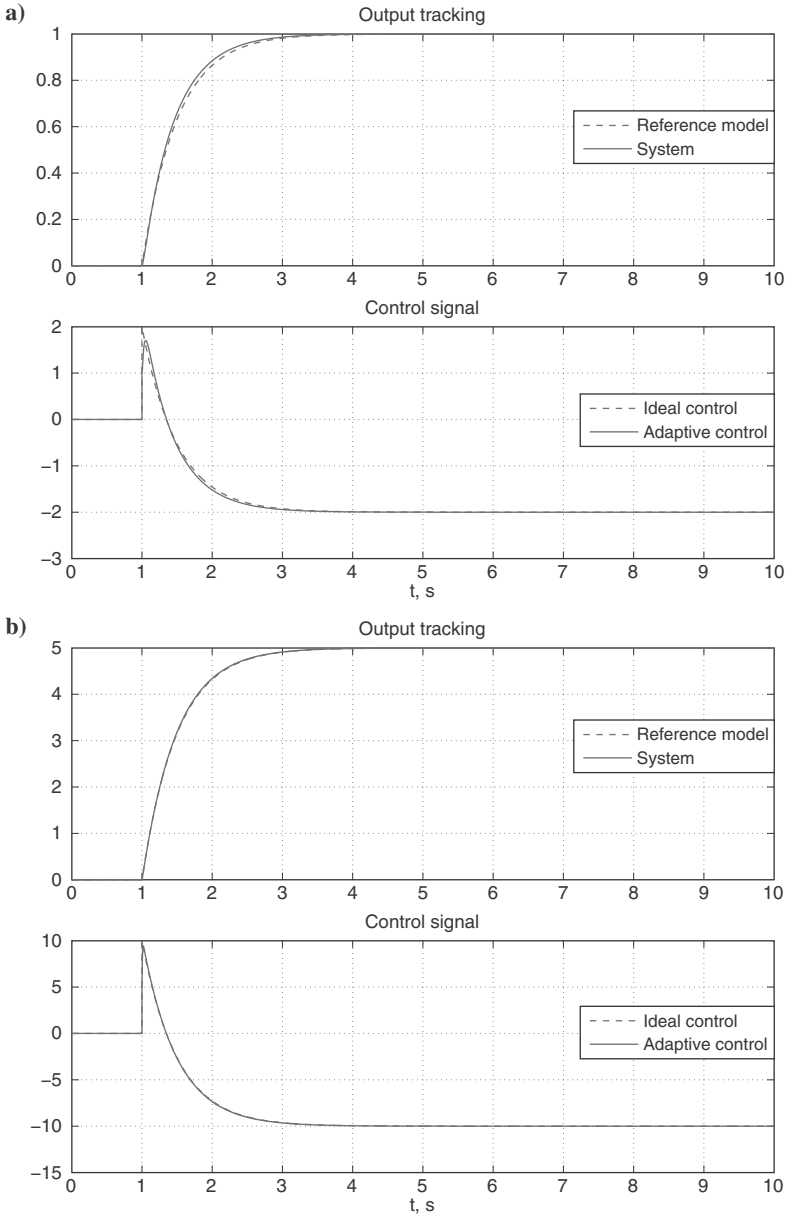


Fig. 4 Step response for large adaptation rates: a) M-MRAC performance for $\gamma = 10,000$ and b) M-MRAC performance for $\gamma = 100,000$.

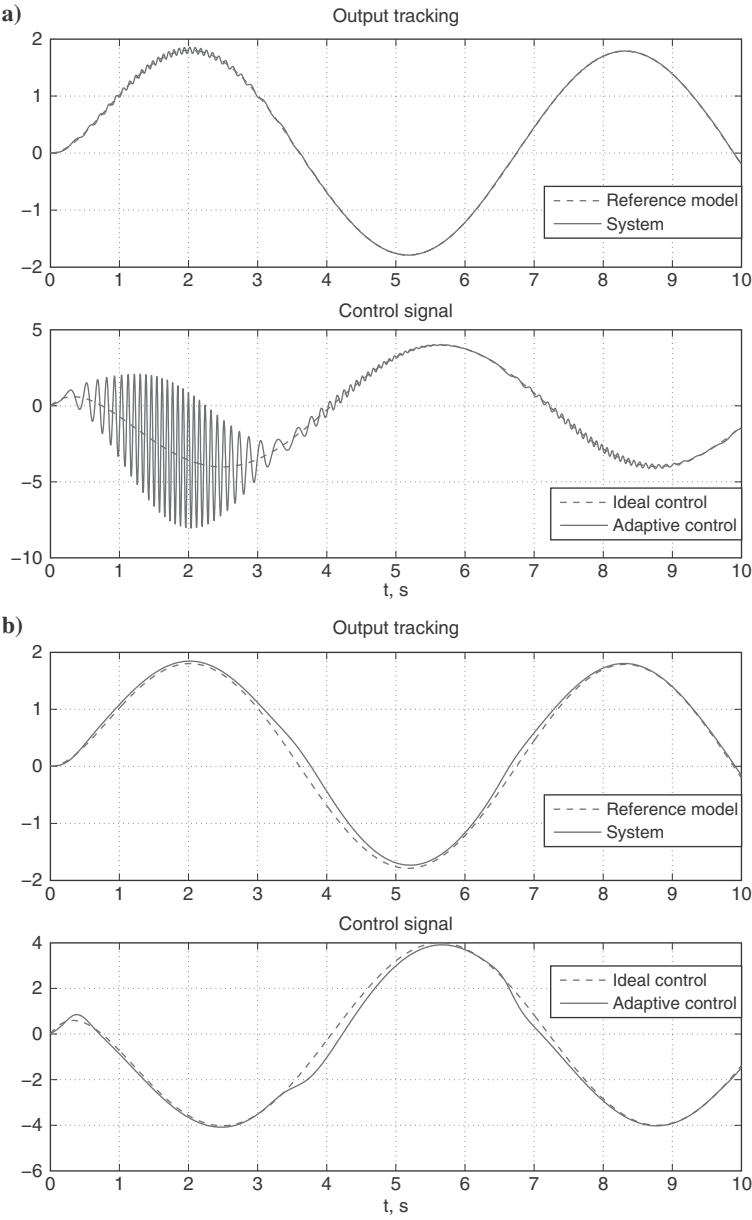


Fig. 5 System's response to sinusoidal command with $\gamma = 1000$: a) MRAC performance and b) M-MRAC performance.

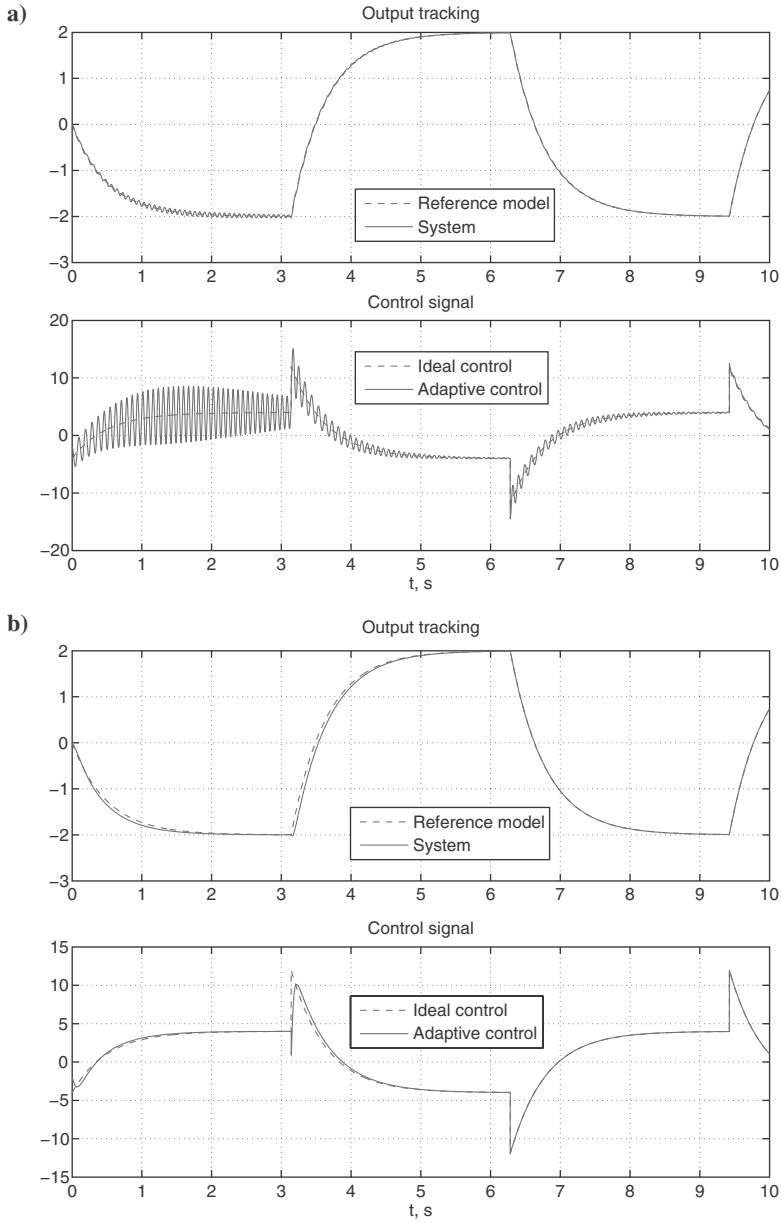


Fig. 6 System's response to square wave command with $\gamma = 1000$: a) MRAC performance and b) M-MRAC performance.

M-MRAC achieves good tracking in both input and output signals without any high-frequency components unlike the MRAC case. The noticeable discrepancies disappear with the increase of γ . In the next set of simulations, we vary the initial conditions to show uniform performance of M-MRAC in initial conditions.

It can be observed from Fig. 7 that M-MRAC not only performs equally well for different initial conditions, but also exhibits scaling properties like linear-time-invariant (LTI) systems.

To show the disturbance rejection properties of M-MRAC, we add an external disturbance $d(t)$ to the system

$$\dot{x}(t) = ax(t) + b[u(t) + d(t)] \quad (96)$$

and switch to the σ -modification type adaptive laws according to Sec. VI. We set $d(t)$ as a unit magnitude square wave of frequency 2 rad/s, and choose $\sigma = 0.3$. Figure 8 displays the system's performance for $\gamma = 1000$ and 10,000. It can be observed that good tracking is achieved in both input and output signals. Moreover, the effect of the disturbance is reduced by increasing γ as predicted in theory.

VIII. AEROSPACE APPLICATION

Now, we apply the M-MRAC design to a dynamic model that represents the lateral-directional motion of a generic transport aircraft (GTM) [20]. The nominal model is the linearized lateral-directional dynamics of GTM at the altitude of 9144 m and speed of 0.8 M and is given by the equation

$$\dot{x}(t) = A_n x(t) + B_n u(t) \quad (97)$$

where $x = [\beta \ r \ p \ \phi]^\top$ is the lateral-directional state vector, in which β is the sideslip angle, r is the yaw rate, p is the roll rate, ϕ is the bank angle, and $u = [\delta_a \ \delta_r]^\top$ is the control signal that includes the aileron deflection δ_a and the rudder deflection δ_r , and the numerical values for A_n and B_n are

$$A_n = \begin{bmatrix} -0.1578 & -0.9907 & 0.0475 & 0.0404 \\ 2.7698 & -0.3842 & 0.0240 & 0 \\ -10.1076 & 0.5090 & -1.7520 & 0 \\ 0 & 0.0506 & 1.0000 & 0 \end{bmatrix}, \quad B_n = \begin{bmatrix} 0.0042 & 0.0476 \\ 0.0351 & -2.2464 \\ 6.3300 & 1.7350 \\ 0 & 0 \end{bmatrix}$$

The controlled output is selected to be the sideslip angle and the bank angle of GTM. The open-loop system has a slow roll mode and negligible damping in the Dutch roll mode (see Table 1).

The reference model is selected from the prospective of improving the performance characteristics of the nominal dynamics. This is achieved by selecting

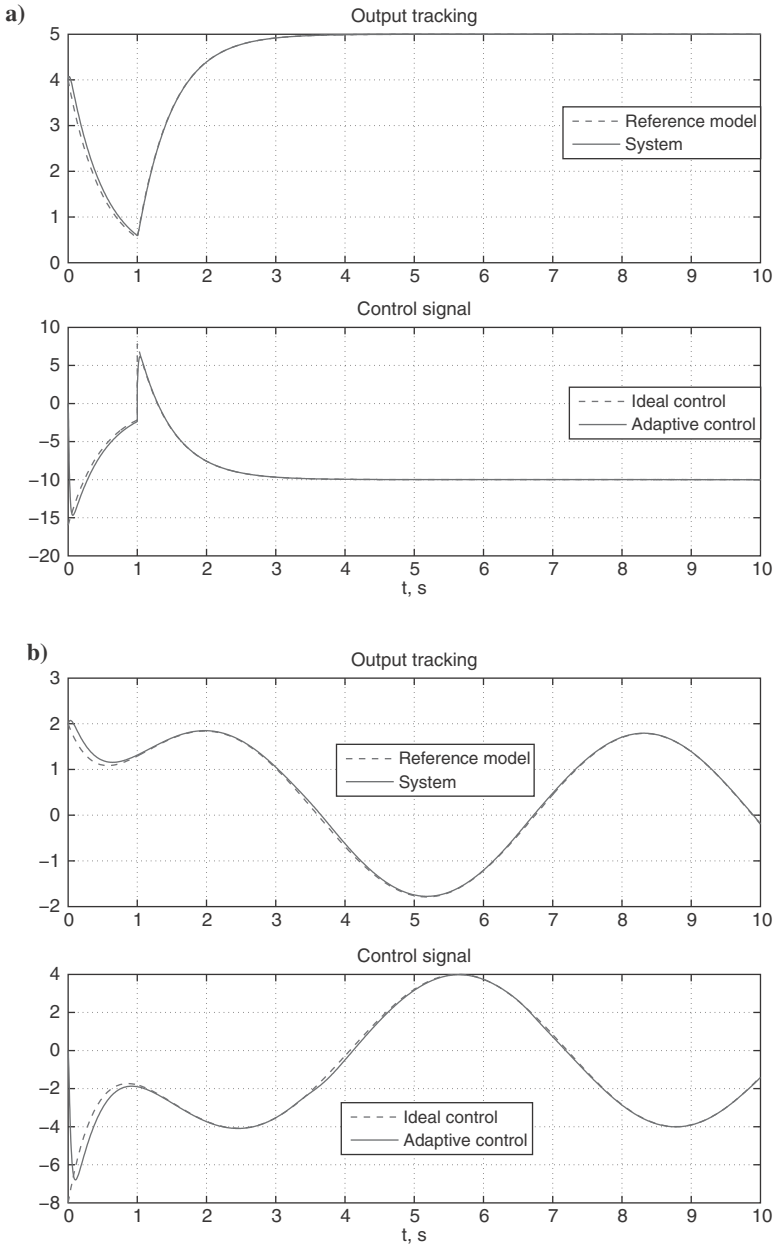


Fig. 7 System's response for different initial conditions with $\gamma = 1000$: a) step response with $x_0 = 4$ and b) sinusoid response with $x_0 = 2$.

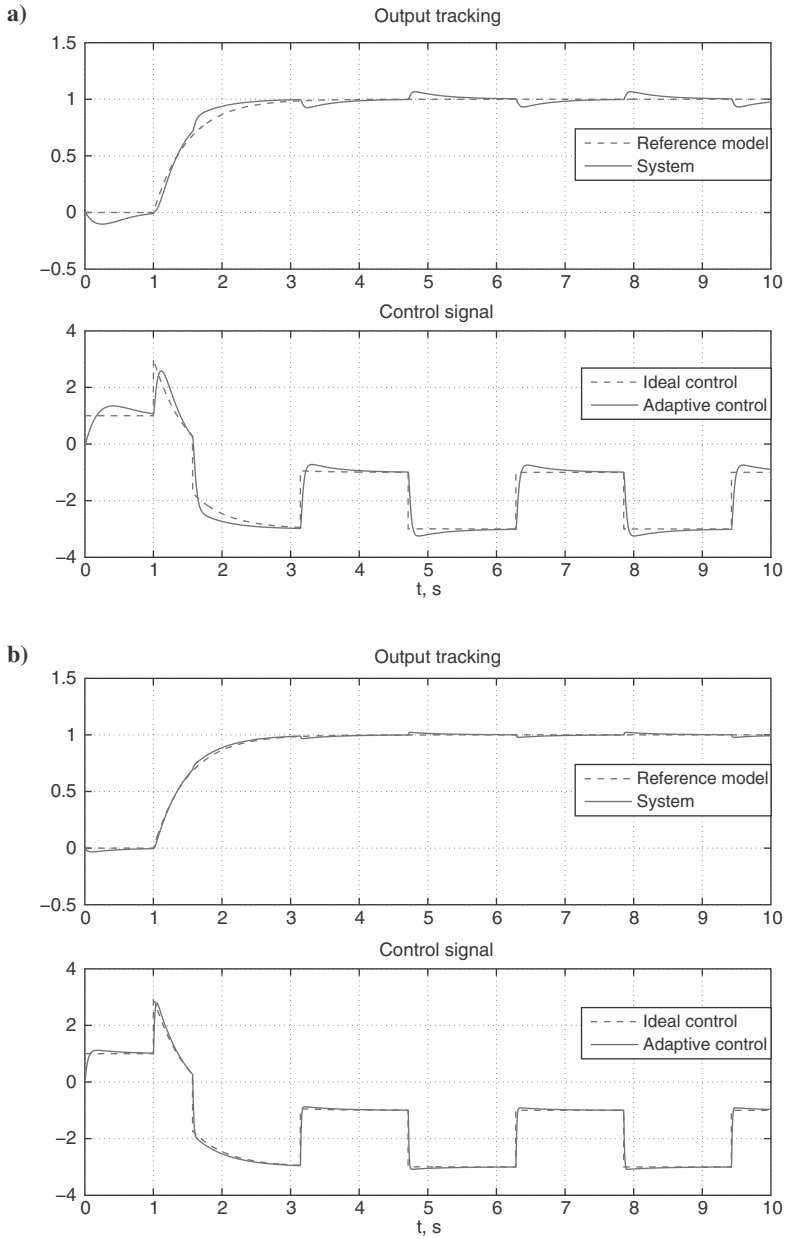


Fig. 8 Step response in the presence of the external disturbance: a) M-MRAC performance with $\gamma = 1000$ and b) M-MRAC performance with $\gamma = 10,000$.

TABLE 1 OPEN-LOOP NOMINAL DYNAMICS CHARACTERISTICS

Eigenvalue	Damping	Natural frequency
-0.0177	1.00	0.0177
-1.6339	1.00	1.63
$-0.3212 \pm 1.7404i$	0.181	1.77

$A_m = A_n - B_n K_0$ and $B_m = B_n$, with the feedback matrix

$$K_0 = \begin{bmatrix} 0 & 0 & 0.4841 & 0.6929 \\ 0.2247 & -1.0966 & 0 & 0 \end{bmatrix}$$

and feedforward matrix

$$N = \begin{bmatrix} 1.2601 & 0.6900 \\ 1.5950 & -0.0512 \end{bmatrix}$$

The selection speeds up the roll mode and increases the damping ratio of the Dutch roll mode. The characteristics of the closed-loop nominal dynamics are presented in Table 2.

The uncertain model of GTM corresponds to about 28% loss of left wing tip and 55% loss of rudder surface. Its dynamics are given by the

$$\dot{\mathbf{x}}(t) = A_m \mathbf{x}(t) + B_m \mathbf{r}(t) + B_m \Lambda [\mathbf{u}(t) - K_1^\top \mathbf{x}(t) - \Lambda^{-\top} \mathbf{r}(t) + \mathbf{d}(t)] \quad (98)$$

where the numerical values of uncertainties are

$$K_1^\top = - \begin{bmatrix} -0.1820 & 0.0149 & -0.1049 & 0 \\ 0.0807 & -0.0109 & 0.0168 & 0 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} 0.5401 & 0.0167 \\ -0.0284 & 0.4736 \end{bmatrix}$$

The control objective is to track a reference command $\mathbf{r}(t) = [\beta_{\text{com}}(t) \phi_{\text{com}}(t)]^\top$ from the zero initial conditions. The reference command is chosen to be a series of coordinated turn maneuvers. That is, $\beta_{\text{com}}(t) = 0$, and the bank

TABLE 2 CLOSED-LOOP NOMINAL DYNAMICS CHARACTERISTICS

Eigenvalue	Damping	Natural frequency
-1.2429	1.00	1.24
-3.4362	1.00	3.44
$-1.5767 \pm 1.1898i$	0.798	1.98

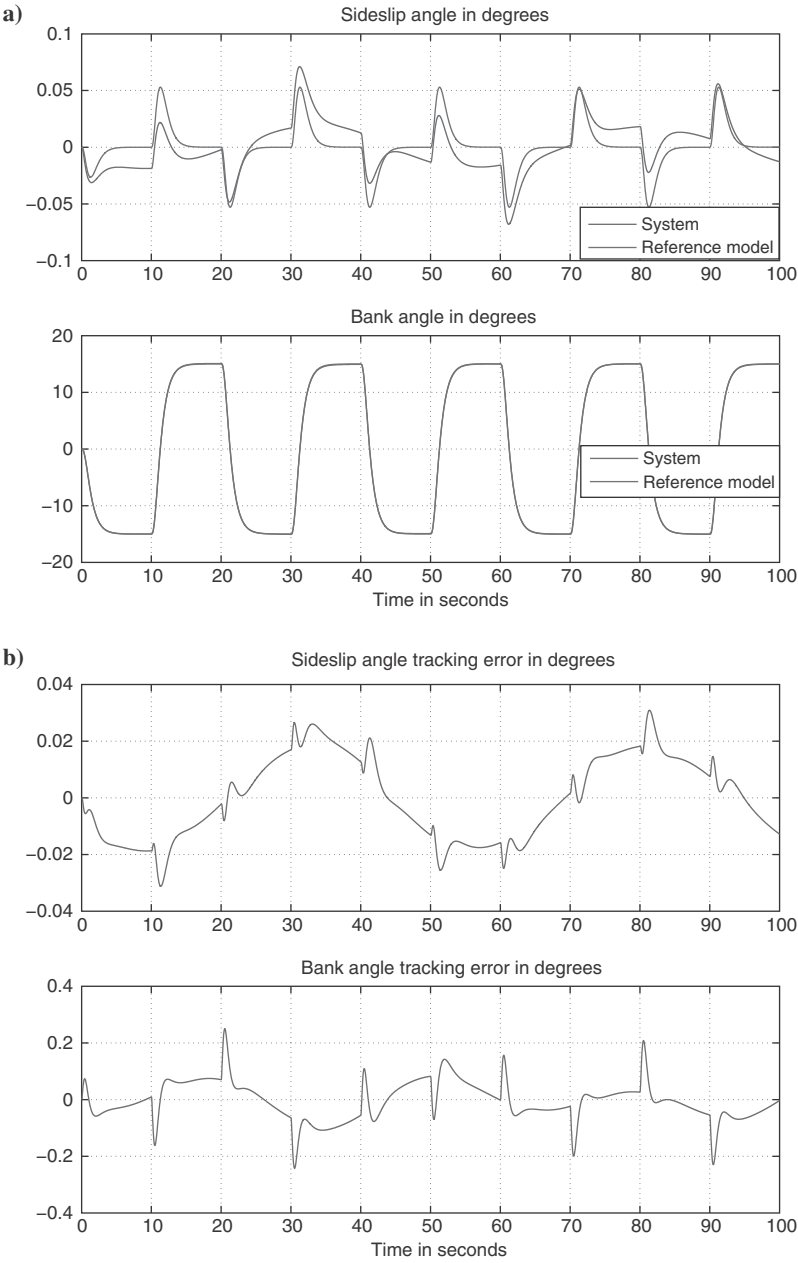


Fig. 9 Output tracking performance of M-MRAC with $\gamma = 600$: a) command tracking and b) tracking error.

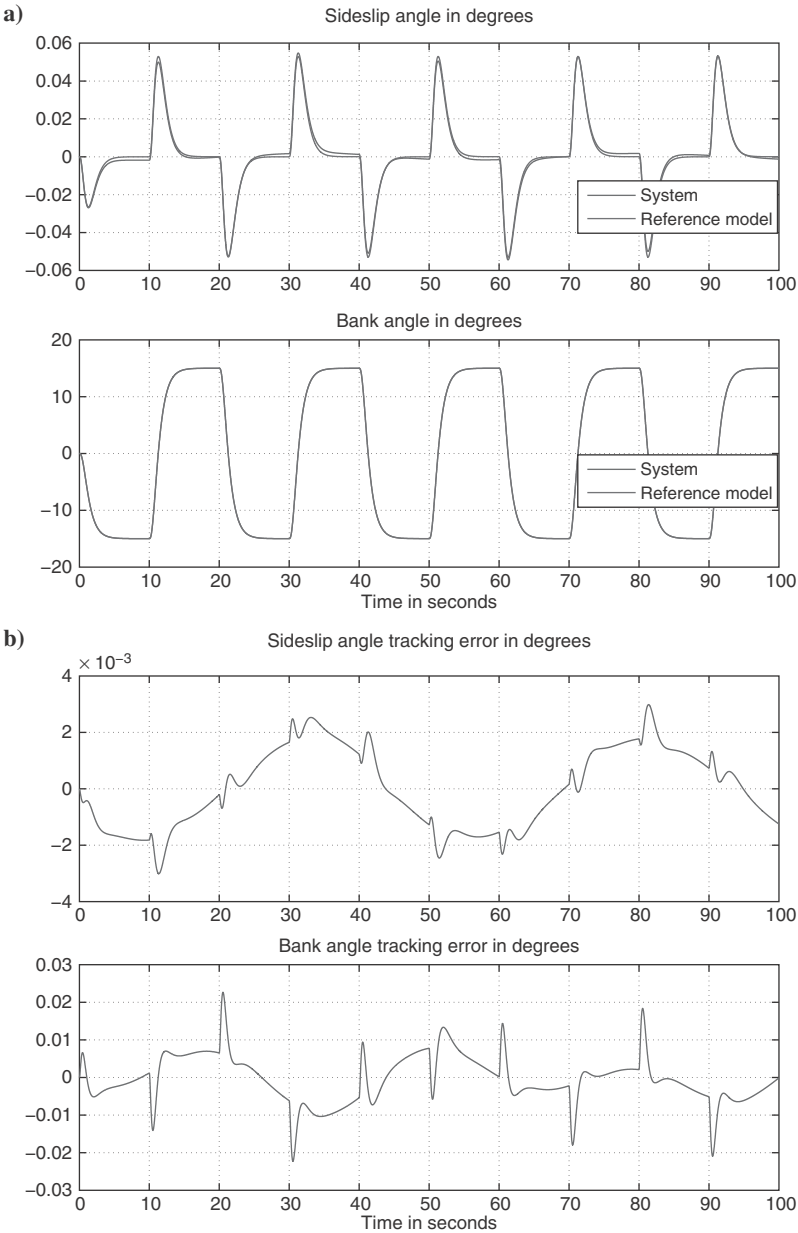


Fig. 11 Output tracking performance of M-MRAC with $\gamma = 60,000$: a) command tracking and b) tracking error.

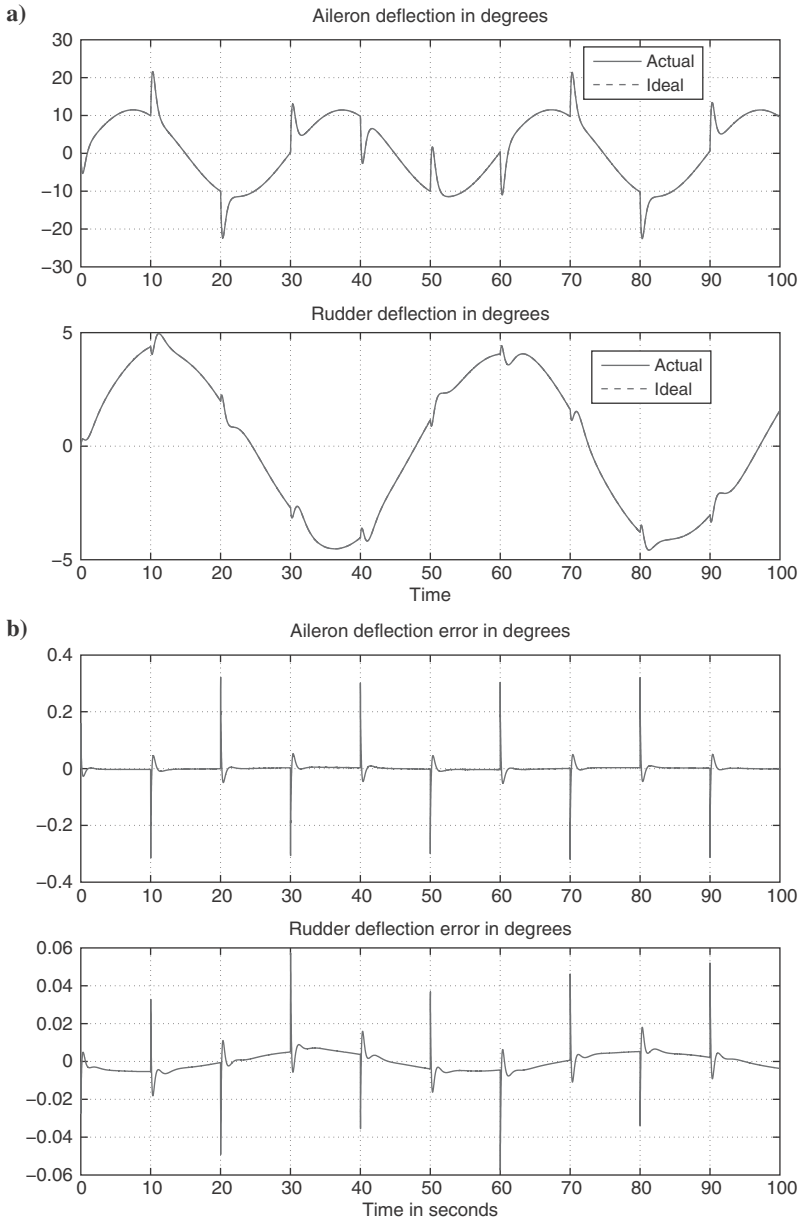


Fig. 12 Input tracking performance of M-MRAC with $\gamma = 60,000$: a) tracking the ideal control signal and b) tracking error.

angle command is chosen to be a square wave of the amplitude of 15 deg and of the frequency $\pi/10$ rad/s. The disturbance is a sinusoidal signal with an amplitude of 0.075 rad and frequency of 0.13 rad/s in the yawing channel and a sinusoidal signal with the amplitude of 0.2 rad and frequency of 0.21 rad/s in the rolling channel. To make $\phi_{\text{com}}(t)$ differentiable, it is filtered through a first-order stable filter $3/(s+3)$. For each γ , parameter λ is selected according to Eq. (56), with $\rho_0 = \pi^2/12^2$ and $s^* = \lambda_{\max}(B_m^T P B_m)$.

First, we run a simulation with $\gamma = 600$ and $Q = \mathbb{I}_4$. Figure 9 displays the command tracking performance and the tracking error of the system, and Fig. 10 displays the ideal control tracking performance and the tracking error. In the bank angle response, the system response and the reference model response are indistinguishable from each other. Clearly, good tracking is achieved in the sideslip angle response as well for the chosen low adaptation rate. The residual tracking error is due to the disturbance signal. In Fig. 10 the ideal control corresponds to the perfect knowledge of the uncertain system. The spikes in the control error are due to initialization errors at the time instances when the square waves change direction, which exponentially dies out as predicted. The sinusoidal shape of the control error corresponds to the disturbance shape.

Next we increase γ 100 times, which corresponds to an increase in λ 10 times. The results are displayed in Figs. 11 and 12. It can be observed that the tracking error is decreased about 10 times as predicted. The error in the input tracking is also decreased approximately in the same proportion, and no high-frequency oscillations are observed.

The results show that good disturbance attenuation can be achieved with the fast adaptation, without generating high-frequency oscillations.

IX. CONCLUSIONS

We have presented a simple yet powerful modification of the conventional MRAC design, which has guaranteed transient and asymptotic performance in both the input and output signals of the system. The proposed M-MRAC design enables one to achieve arbitrarily close tracking of ideal output and input signals by increasing the adaptation rate without generating high-frequency oscillations in the control signal. The performance of M-MRAC is uniform for various initial conditions and in reference commands.

Because only the reference model is modified, this method can be applied along with any known robust modification scheme to prevent parameter drift, when the system is subject to disturbances. The analysis of the transient properties of M-MRAC is provided in the case of σ modification. It is shown analytically and demonstrated in the simulation examples that the systems' input and output tracking errors can be decreased as desired by increasing the adaptation rate in the presence of bounded external disturbances.

REFERENCES

- [1] Zang, Z., and Bitmead, R., "Transient Bounds for Adaptive Control Systems," *Proceedings of the 29th IEEE Conference on Decision and Control*, 1990, pp. 2724–2729.
- [2] Datta, A., and Ho, M.-T., "On Modifying Model Reference Adaptive Control Schemes for Performance Improvement," *IEEE Transactions on Automatic Control*, Vol. 39, No. 9, 1994, pp. 1977–1980.
- [3] Datta, A., and Ioannou, P., "Performance Analysis and Improvement in Model Reference Adaptive Control," *IEEE Transactions on Automatic Control*, Vol. 39, No. 12, 1994, pp. 2370–2387.
- [4] Sun, J., "A Modified Model Reference Adaptive Control Scheme for Improved Transient Performance," *IEEE Transactions on Automatic Control*, Vol. 38, No. 8, 1993, pp. 1255–1259.
- [5] Morse, S., "Supervisory Control of Families of Linear Set-Point Controllers—Part 1: Exact Matching," *IEEE Transactions on Automatic Control*, Vol. 41, No. 10, 1996, pp. 1413–1431.
- [6] Morse, S., "Supervisory Control of Families of Linear Set-Point Controllers—Part 2: Robustness," *IEEE Transactions on Automatic Control*, Vol. 42, No. 11, 1997, pp. 1500–1515.
- [7] Arteaga, A. M., and Tang, Y., "Adaptive Control of Robots with an Improved Transient Performance," *IEEE Transactions on Automatic Control*, Vol. 47, No. 7, 2002, pp. 1198–1202.
- [8] Krstic, M., Kanellakopoulos, I., and Kokotovic, P., *Nonlinear and Adaptive Control Design*, Wiley, New York, 1995.
- [9] French, M., Szepesvari, C., and Rogers, E., "Uncertainty, Performance, and Model Dependency in Approximate Adaptive Nonlinear Control," *IEEE Transactions on Automatic Control*, Vol. 45, No. 2, 2000, pp. 353–358.
- [10] Cao, C., and Hovakimyan, N., "Design and Analysis of a Novel \mathcal{L}_1 Adaptive Control Architecture with Guaranteed Transient Performance," *IEEE Transactions on Automatic Control*, Vol. 53, No. 2, 2008, pp. 586–591.
- [11] Stepanyan, V., Krishnakumar, K., and Nguyen, N., "Guaranteed Input-Output Transient Performance with Filtering Robust Adaptive Control," *IEEE Transactions on Neural Networks* (submitted for publication).
- [12] Bernstein, D. S., *Matrix Mathematics*, Princeton Univ. Press, Princeton, NJ, 2005, p. 380.
- [13] Khalil, H., *Nonlinear Systems*, 3rd ed., Prentice–Hall, Upper Saddle River, NJ, 2002, p. 199.
- [14] Rugh, W. J., *Linear System Theory*, Prentice–Hall, Upper Saddle River, NJ, 1995, p. 135.
- [15] Narendra, K., and Annaswamy, A., *Stable Adaptive Control*, Prentice–Hall, Upper Saddle River, NJ, 1989.
- [16] Zhang, F., and Leonard, N. E., "Coordinated Patterns of Unit Speed Particles on a Closed Curve," *Systems and Control Letters*, Vol. 56, No. 6, 2007, pp. 397–407.
- [17] Sastry, S. S., and Bodson, M., *Adaptive Control: Stability, Convergence and Robustness*, Prentice–Hall, Upper Saddle River, NJ, 1989, p. 19.

- [18] Coddington, E. A., and Levinson, N., *Theory of Ordinary Differential Equations*, McGraw-Hill, New York, 1955, p. 42.
- [19] Narendra, K., and Annaswamy, A., “A New Adaptive Law for Robust Adaptation Without Persistent Excitation,” *IEEE Transactions on Automatic Control*, Vol. 32, No. 2, 1987, pp. 134–145.
- [20] Nguyen, N., Krishnakumar, K., Kaneshige, J., and Nespeca, P., “Flight Dynamics and Hybrid Adaptive Control of Damaged Aircraft,” *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 3, 2008, pp. 751, 764.

\mathcal{L}_1 Adaptive Control in Flight

Enric Xargay* and Naira Hovakimyan[†]

University of Illinois at Urbana–Champaign, Urbana, Illinois 61801

Vladimir Dobrokhodov[‡] and Isaac Kaminer[§]

Naval Postgraduate School, Monterey, California 93943

Chengyu Cao[¶]

University of Connecticut, Storrs, Connecticut 06269

Irene M. Gregory^{**}

NASA Langley Research Center, Hampton, Virginia 23681

I. FAST ADAPTATION: THE KEY TO SAFE FLIGHT

ADAPTIVE control has long been seen as an appealing technology with the potential to improve aircraft performance in challenging flight conditions and increase aviation safety in the event of severe unexpected failures or vehicle damage. However, several limitations of the conventional adaptive systems have prevented this technology from being widely used in safety-critical aerospace applications. Incomplete overviews on fundamental deficiencies of adaptive control are provided in [1, 2]. Additionally, discussions on open problems and certification challenges for adaptive flight control systems can be found in [3–5]. In particular, the key deficiencies of conventional adaptive (flight) control systems can be summarized as 1) the lack of predictability of the closed-loop response, 2) the limited framework for analysis of the robustness and performance guarantees of closed-loop adaptive systems, and 3) the lack of systematic design guidelines to solve the tradeoff between adaptation, performance, and robustness.

These limitations seem to be directly related to the inability of conventional adaptive control architectures to *separate* the timescale of the adaptation (or identification) process from the timescales of the closed-loop dynamics and the plant uncertainty dynamics. If these timescales are comparable, interaction of the three processes can occur and lead to closed-loop instability. An apparent consequence

*Ph.D. Candidate, Department of Aerospace Engineering; xargay@illinois.edu.

[†]Professor, Department of Mechanical Science and Engineering; nhovakim@illinois.edu.

[‡]Research Associate Professor, Department of Mechanical and Astronautical Engineering; vldobr@nps.edu.

[§]Professor, Department of Mechanical and Astronautical Engineering; kaminer@nps.edu.

[¶]Assistant Professor, Department of Mechanical Engineering; ccao@engr.uconn.edu.

^{**}Senior Aerospace Research Engineer, Dynamic Systems and Controls Branch, MS 308; irene.m.gregory@nasa.gov.

of this architectural limitation is the *asymptotic* nature of the results obtained in the development of the theory of adaptive control over the years. In fact, when dealing with safety-critical systems (such as flight control systems), features such as boundedness, ultimate boundedness, or even asymptotic convergence are *weak* properties for closed-loop feedback systems. Much stronger guarantees are needed. On one hand, performance requirements demand predictable and consistent response of the closed-loop system, dependent upon changes in system dynamics and reference signals. On the other hand, system uncertainty requires accurate quantification of the robustness and the stability margins of the feedback loop.

The importance of transient performance guarantees in flight control can be illustrated through Fig. 1, which presents loss-of-control accident data relative to angle of attack α and angle of sideslip β . The dark-grey region for angle of attack and sideslip represents the normal flight envelope, where an aircraft usually flies in the absence of abnormalities. The mid-grey area represents configurations for which high-fidelity nonlinear aerodynamic models of the aircraft are available from wind-tunnel data. Outside this wind-tunnel data envelope, the aerodynamic models available are typically obtained by extrapolating wind-tunnel test data, and hence are highly uncertain. This fact suggests that pilots might not be adequately trained to fly the aircraft in these regimes, or, in the case of autonomous vehicles, the guidance and stabilization loops might not be properly designed for safe recovery. Moreover, it is not reasonable to rely on a flight control system to compensate for the uncertainty in these flight conditions because aircraft dynamic controllability is not guaranteed in such regimes. Therefore, from a *safety standpoint*, the main objective of a flight control system is to ensure that an aircraft, suddenly experiencing an unexpected failure, does not “escape” its α - β wind-tunnel data envelope, provided that enough control authority remains. In this

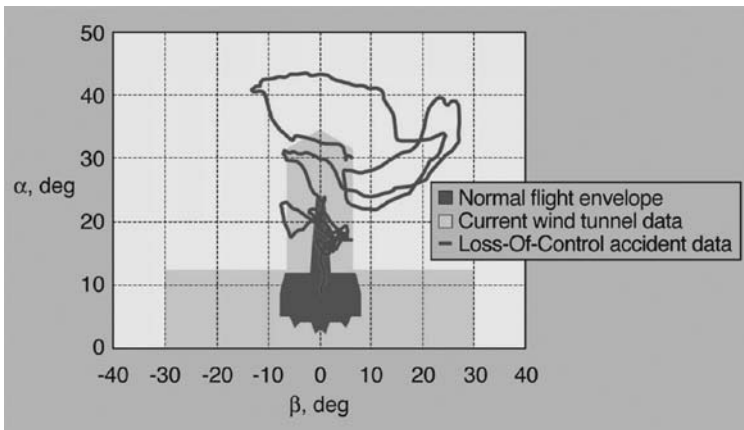


FIG. 1 Loss-of-control accident data relative to angle of attack α and angle of sideslip β (from [6]).

sense, (adaptive) flight control systems with only asymptotic guarantees might not prevent the aircraft from entering adverse flight conditions with unusual attitudes. Aircraft safe operation and recovery clearly require flight control architectures that ensure a predictable transient response of the impaired aircraft, by quickly adapting to the failure. This, of course, demands *fast estimation schemes* with high adaptation rates.

The theory of \mathcal{L}_1 adaptive control overcomes the limitations of conventional adaptive control architectures just described and enables the design of robust adaptive control architectures using fast adaptation schemes [7, 8]. The key feature of \mathcal{L}_1 adaptive control is the decoupling of the adaptation loop from the control loop, which enables fast adaptation without sacrificing robustness. In this context, *fast adaptation* indicates that the adaptation rate in \mathcal{L}_1 architectures is to be selected so that the timescale of the adaptation process is faster than the timescales associated with plant parameter variations and underlying closed-loop dynamics. In fact, in \mathcal{L}_1 adaptive control architectures the rate of the adaptation loop can be set arbitrarily high, subject only to hardware limitations—computational power and high-frequency sensor noise—while the tradeoff between performance and robustness can be addressed through conventional methods from classical and robust control. This separation between adaptation and robustness is achieved by explicitly building the robustness specification into the problem formulation, with the understanding that the uncertainties in any feedback loop can be compensated for only within the bandwidth of the control channel. From an architectural perspective, this modification of the problem formulation leads to the insertion of a bandwidth-limited filter in the feedback path, which ensures that the control signal stays in the desired frequency range. On one hand, the fact that the adaptation rate can be arbitrarily high allows for compensation of the undesirable effects of rapidly varying uncertainties and significant changes in the system dynamics. Fast adaptation is also critical to achieve predictable transient performance for both of the system's signals, input and output, without enforcing persistency of excitation or resorting to high-gain feedback. On the other hand, the bandwidth-limited filter keeps the robustness margins bounded away from zero in the presence of fast adaptation. To this extent, the bandwidth and the structure of this filter define the tradeoff between performance and robustness of the closed-loop adaptive system. These features of \mathcal{L}_1 adaptive control have been verified—*consistently with the theory*—in a large number of flight tests and mid- to high-fidelity simulation environments (for example, see [9–16]).

This chapter presents flight-test results that demonstrate the advantages of \mathcal{L}_1 adaptive control as a predictable robust adaptive control architecture with the potential to provide enhanced stability and maneuverability margins for safe aircraft operation in adverse conditions. The chapter is organized as follows. Section II presents flight-test results of an unmanned aerial vehicle augmented with an \mathcal{L}_1 flight control system. Section III describes the flight tests on the NASA AirSTAR flight-test vehicle. Finally, Sec. IV summarizes the key results and contains the main conclusions as well as directions for future research.

II. \mathcal{L}_1 ADAPTIVE CONTROL FOR THE NPS AUTONOMOUS UAV

Recognizing the value of experimental verification and validation of advanced flight control algorithms, the Naval Postgraduate School (NPS) has developed the Rapid Flight Test Prototyping System (RFTPS) [17]; see Fig. 2. The RFTPS consists of a testbed SIG Rascal unmanned aerial vehicle (UAV) equipped with a commercial autopilot, an embedded computer running research algorithms in real time, and a ground control station for flight management and data monitoring/collection. This system facilitates real-time onboard integration of advanced control algorithms and provides the opportunity to design and conduct comprehensive flight-test programs to evaluate the robustness and performance characteristics of these algorithms.

To easily accommodate changing payload configurations and improve the angular-rate tracking capabilities of the UAV in the event of control surface failures, the commercial autopilot of the RFTPS has been augmented with an \mathcal{L}_1 adaptive output-feedback architecture. The inner-loop \mathcal{L}_1 flight control architecture implemented on the RFTPS is represented in Fig. 3. A brief overview of the \mathcal{L}_1 output-feedback control law is presented next, together with a set of representative results from the extensive flight-test program conducted by NPS since 2006 in Camp Roberts, California. The reader will find detailed explanations, hardware-in-the-loop simulations, and further flight-test results in [18–22].

A. \mathcal{L}_1 ADAPTIVE OUTPUT-FEEDBACK CONTROLLER FOR AUTOPILOT AUGMENTATION

The adaptive autopilot augmentation implemented on the SIG Rascal UAV is an \mathcal{L}_1 output-feedback control architecture for strictly positive real reference systems. A brief review of this control architecture is presented next for the case of first-order reference systems. Details on the adaptive output-feedback control law, along with the main theoretical guarantees, can be found in Sec. 4.1 of [7].

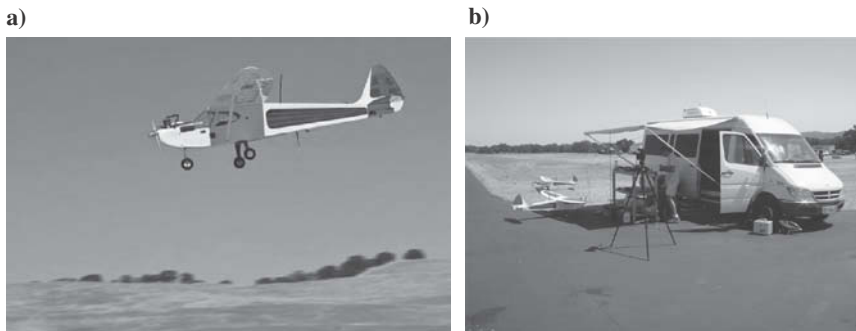


FIG. 2 Rapid Flight Test Prototyping System at the Naval Postgraduate School: a) SIG Rascal 110 research aircraft and b) ground control station.

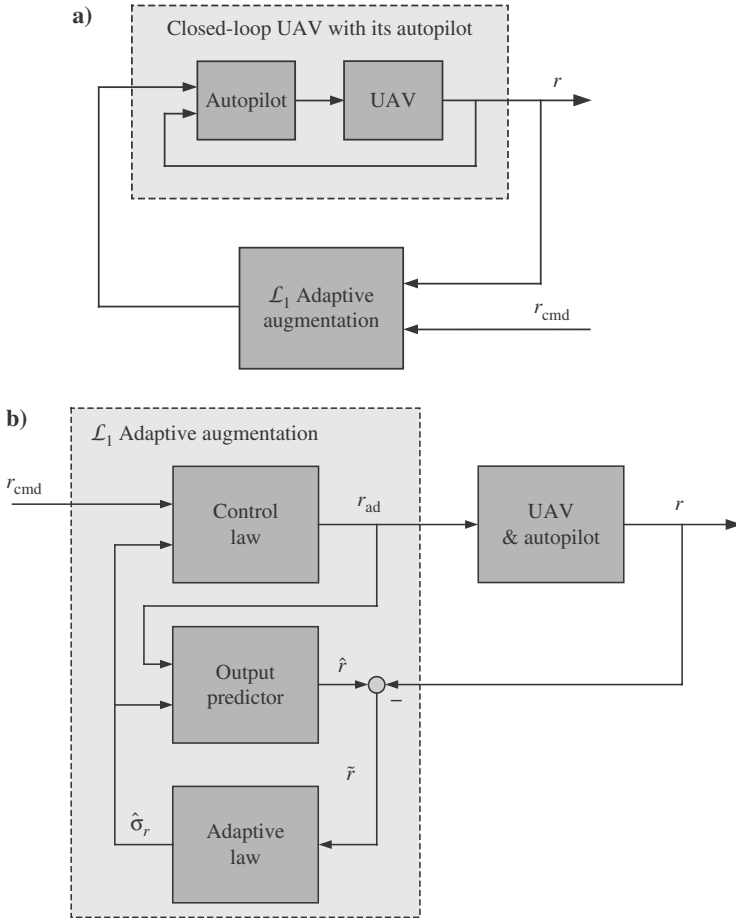


FIG. 3 Inner-loop \mathcal{L}_1 augmentation loop tested by NPS: a) inner-loop structure with the \mathcal{L}_1 augmentation loop and b) \mathcal{L}_1 controller for turn-rate control.

Consider the single-input, single-output plant

$$y(s) = A(s)[u(s) + d(s)] \quad (1)$$

where $u(t) \in \mathbb{R}$ is the control input; $y(t) \in \mathbb{R}$ is the output; $A(s)$ is a strictly proper unknown transfer function of unknown relative degree; and $d(s)$ is the Laplace transform of the time- and output-dependent uncertainty $d(t) = f[t, y(t)]$. It is assumed that there exist constants $L > 0$ and $L_0 > 0$ such

that, for all time $t \geq 0$ and all y, y_1, y_2

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|, \quad |f(t, y)| \leq L|y| + L_0 \quad (2)$$

The desired closed-loop behavior is given in terms of the first-order stable transfer function

$$M(s) = \frac{m}{s + m} \quad m > 0$$

The control objective is thus to design an output-feedback controller $u(t)$ such that the system output $y(t)$ tracks bounded piecewise continuous reference signals $r(t)$ following the preceding reference system, both in *transient* and *steady state*.

The plant (1) can be rewritten in terms of the desired system as

$$y(s) = M(s)[u(s) + \sigma(s)] \quad (3a)$$

$$\sigma(s) = \frac{[A(s) - M(s)]u(s) + A(s)d(s)}{M(s)} \quad (3b)$$

where the uncertainties due to $A(s)$ and $d(t)$ have been lumped into the signal $\sigma(t)$. The philosophy of the \mathcal{L}_1 output-feedback controller is to obtain an estimate of the uncertain signal $\sigma(t)$ and define a control signal that compensates for the effects of this uncertainty within the bandwidth of a low-pass filter $C(s)$ introduced in the control channel. This filter leads to separation between adaptation and robustness, guarantees that the output of the \mathcal{L}_1 controller stays in the low-frequency range in the presence of fast adaptation, and characterizes the tradeoff between performance and robustness. Adaptation is based on the projection operator, ensuring boundedness of the adaptive parameters by definition [23], and uses the output of an output predictor to update the estimate of the uncertainty $\hat{\sigma}(t)$. This output predictor is defined to have the same structure as the system in Eq. (3a) but using the estimate $\hat{\sigma}(t)$ instead of $\sigma(t)$ itself, which is unknown. The \mathcal{L}_1 adaptive control architecture is represented in Fig. 3b, and its elements are introduced next.

The *output predictor* is given by

$$\dot{\hat{y}}(t) = -m\hat{y}(t) + m[u(t) + \hat{\sigma}(t)] \quad \hat{y}(0) = 0 \quad (4)$$

where $\hat{\sigma}(t)$ is the adaptive estimate. The *adaptation law* for $\hat{\sigma}(t)$ is defined as

$$\dot{\hat{\sigma}}(t) = \Gamma \text{Proj}[\hat{\sigma}(t), -\tilde{y}(t)] \quad \hat{\sigma}(0) = 0 \quad (5)$$

where $\tilde{y}(t) \triangleq \hat{y}(t) - y(t)$ is the output-prediction error signal, $\Gamma > 0$ is the adaptation rate, and $\text{Proj}(\cdot, \cdot)$ denotes the Pomet–Praly projection operator [23]. Finally, the *control signal* is generated as

$$u(s) = r(s) - C(s)\hat{\sigma}(s) \quad (6)$$

where $C(s)$ is a strictly proper low-pass filter with $C(0) = 1$.

The ideal (nonimplementable) control signal for this problem is given by

$$u_{id}(t) = r(t) - \sigma(t)$$

which corresponds to the input that yields the desired system response

$$y_{id}(s) = M(s)r(s)$$

by exactly canceling the uncertainty. A filtered version of this ideal control law [which aims at compensating for the uncertainty only within the bandwidth of the filter $C(s)$] is now used to define the auxiliary \mathcal{L}_1 reference system

$$y_{ref}(s) = M(s)[u_{ref}(s) + \sigma_{ref}(s)] \quad (7a)$$

$$u_{ref}(s) = r(s) - C(s)\sigma_{ref}(s) \quad (7b)$$

where

$$\sigma_{ref}(s) \triangleq \frac{[A(s) - M(s)]u_{ref}(s) + A(s)d_{ref}(s)}{M(s)}$$

with $d_{ref}(s)$ being the Laplace transform of $d_{ref}(t) \triangleq f[t, y_{ref}(t)]$. This auxiliary \mathcal{L}_1 reference system is defined thus as the nonadaptive version of the \mathcal{L}_1 adaptive controller assuming perfect knowledge of uncertainties and is used to characterize the performance of the closed-loop adaptive system.

The next theorem summarizes the main result for this adaptive architecture.

Theorem 1 ([7] Theorem 4.1.1)

Assume that the low-pass filter $C(s)$ for the given $M(s)$ is chosen such that the transfer function

$$H(s) \triangleq \frac{A(s)M(s)}{C(s)A(s) + [1 - C(s)]M(s)}$$

is stable and satisfies the \mathcal{L}_1 -norm condition

$$\|G(s)\|_{\mathcal{L}_1} L < 1 \quad (8)$$

where $G(s) \triangleq H(s)[1 - C(s)]$ and L was introduced in Eq. (2). Then, the \mathcal{L}_1 reference system in Eq. (7) is stable. Furthermore, there exists an adaptation rate Γ such that

$$\|\tilde{y}\|_{\mathcal{L}_\infty} \leq \gamma_0 \left(1/\sqrt{\Gamma}\right)$$

$$\|y_{ref} - y\|_{\mathcal{L}_\infty} \leq \gamma_1 \left(1/\sqrt{\Gamma}\right)$$

$$\|u_{ref} - u\|_{\mathcal{L}_\infty} \leq \gamma_2 \left(1/\sqrt{\Gamma}\right)$$

where $\gamma_0()$, $\gamma_1()$, and $\gamma_2()$ are known class \mathcal{K} functions.

The preceding bounds imply that the tracking error between $y(t)$ and $y_{\text{ref}}(t)$, as well as between $u(t)$ and $u_{\text{ref}}(t)$, are uniformly bounded by class \mathcal{K} functions of $1/\sqrt{\Gamma}$. This fact implies that, during the transient phase, it is possible to achieve arbitrary improvement of tracking performance by uniformly increasing the adaptation rate Γ .

B. \mathcal{L}_1 ADAPTATION FOR PATH-FOLLOWING MISSIONS

Conventional autopilots are normally designed to provide only guidance loops for waypoint navigation. To extend the range of possible applications of (small) UAVs equipped with traditional autopilots, solutions to the problem of three-dimensional path-following control have been presented in [21] and [24]. The proposed solutions exhibit a multiloop control structure, with an outer-loop path-following control law that relies on a nonlinear control strategy derived at the kinematic level and an inner loop consisting of the commercial autopilot augmented with the \mathcal{L}_1 controller. The overall closed-loop system with the \mathcal{L}_1 augmentation loop is presented in Fig. 4.

Both the autopilot and the path-following guidance loop are tuned for a nominal aircraft configuration. When the aircraft configuration changes due to, for instance, the presence of a different payload, the performance of the closed-loop UAV with its autopilot can deteriorate significantly. Flight-test results for one of these off-nominal payload conditions are presented next. In particular, flight-test results comparing the performance of the path-following algorithm with and without \mathcal{L}_1 adaptation are shown in Fig. 5. The flight-test data include the two-dimensional horizontal projection of the commanded and the actual paths, the commanded $r_c(t)$ and the measured $r(t)$ turn-rate responses, and the path-following errors $y_F(t)$ and $z_F(t)$. The results show that the UAV is

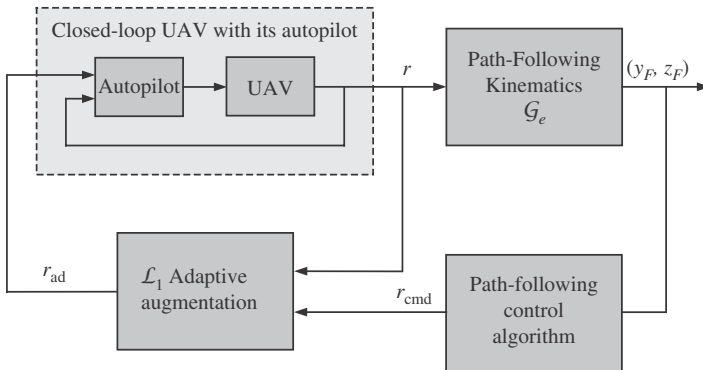


FIG. 4 Closed-loop path-following system with the \mathcal{L}_1 augmentation loop.

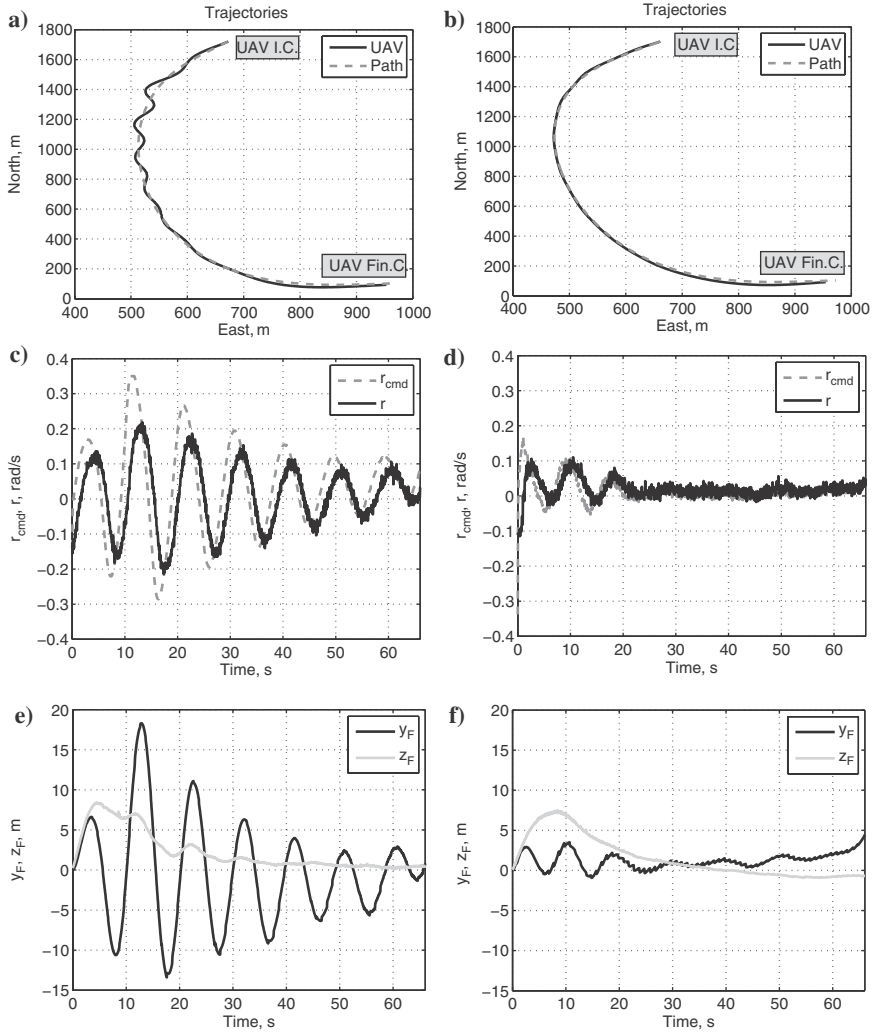


FIG. 5 Path-following performance with and without \mathcal{L}_1 augmentation loop: a) \mathcal{L}_1 OFF: two-dimensional projection, b) \mathcal{L}_1 ON: two-dimensional projection, c) \mathcal{L}_1 OFF: turn rate, d) \mathcal{L}_1 ON: turn rate, e) \mathcal{L}_1 OFF: path-following errors, and f) \mathcal{L}_1 ON: path-following errors.

able to follow the path, keeping the path-following errors reasonably small during the whole experiment. The plots also demonstrate the improved path-following performance when the \mathcal{L}_1 augmentation loop is engaged. On one hand, one can observe that the nominal outer-loop path-following controller exhibits

significant oscillatory behavior, with rate commands going up to 0.35 rad/s and with path-following errors around 18 m. On the other hand, the \mathcal{L}_1 augmentation loop is able to improve the angular-rate tracking capabilities of the inner-loop controller, which results in rate commands not exceeding 0.15 rad/s and path-following errors below 8 m. Furthermore, it is important to note that the adaptive controller does not introduce any high-frequency content into the commanded turn-rate signal, as it can be seen by comparing Figs. 5c and 5d.

Finally, it is important to mention that the benefits of the \mathcal{L}_1 augmentation loop are not limited to practical flight-test results, but also extend to theoretical claims. In the derivations in [21], the uniform performance bounds that the \mathcal{L}_1 controller guarantees both in transient and steady state are critical to prove stability of the path-following closed-loop system, which takes into account the dynamics of the UAV with its autopilot.

C. \mathcal{L}_1 ADAPTATION IN THE PRESENCE OF CONTROL SURFACE FAILURES

The path-following flight-test setup introduced in Sec. II.A was used in [18] to demonstrate that the \mathcal{L}_1 augmentation loop provides fast recovery to sudden locked-in-place failures in either one of the ailerons or in the rudder of the RFTPS, while the nominal unaugmented system goes unstable. In these experiments, the extended capabilities of the RFTPS were used to instantaneously deflect and hold preprogrammed combinations of control surfaces at a predefined position without notifying or reconfiguring the commercial autopilot.

Although the flight experiments considered failures in the left aileron covering the range from 0 to -12 deg (with respect to a trim value -2.34 deg), and rudder failures from 0 to 2 deg, this section presents only an extract from these results. In particular, Figs. 6 and 7 illustrate the performance of the path-following system with two levels of sudden left-aileron locked-in-place failures at -2 deg and -10 deg (with respect to trim value -2.34 deg):

1. Analysis of the -2 -deg case showed that even such a small deflection pushes the UAV away to the right from the desired path (see Fig. 6a), resulting in almost 25 m of lateral cross-track error (see Fig. 6b). After the failure is introduced, the UAV converges to a 15-m lateral error boundary in about 20 s.
2. The results of the -10 deg left-aileron failure (Fig. 7) are similar to the ones for the preceding case. Naturally, the errors and the control efforts increase due to the increased severity of the failure, and the impaired UAV converges to the 15-m boundary in approximately 45 s.

Moreover, analysis of the entire series of results with left-aileron failures (covering the range from 0 to -12 deg) shows a *graceful* degradation in the path-following performance.

The preceding results demonstrate that integration of an \mathcal{L}_1 augmentation loop provides fault-tolerance capabilities to the autonomous vehicle, by automatically readjusting the control signals to stabilize the impaired airplane and using the remaining control authority to steer the airplane along the path. This is done without resorting to fault detection and isolation methods or reconfiguration of the existing inner-loop control structure.

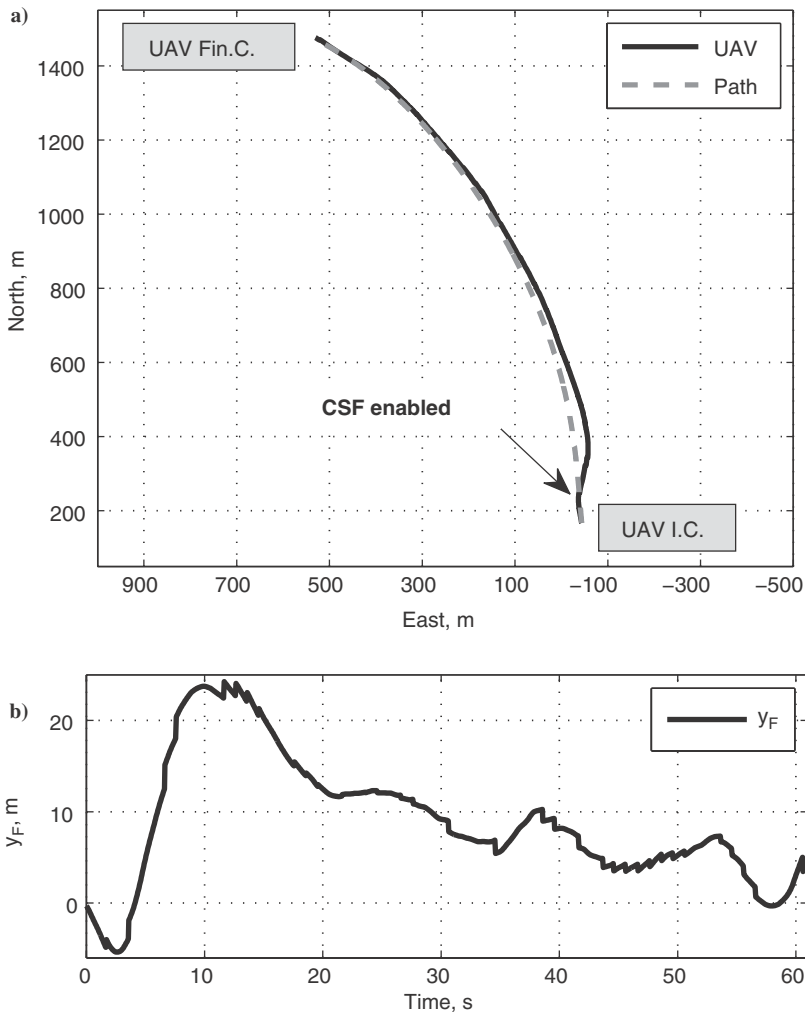


FIG. 6 Two-deg locked-in-place left-aileron failure: a) two-dimensional projection and b) lateral error.

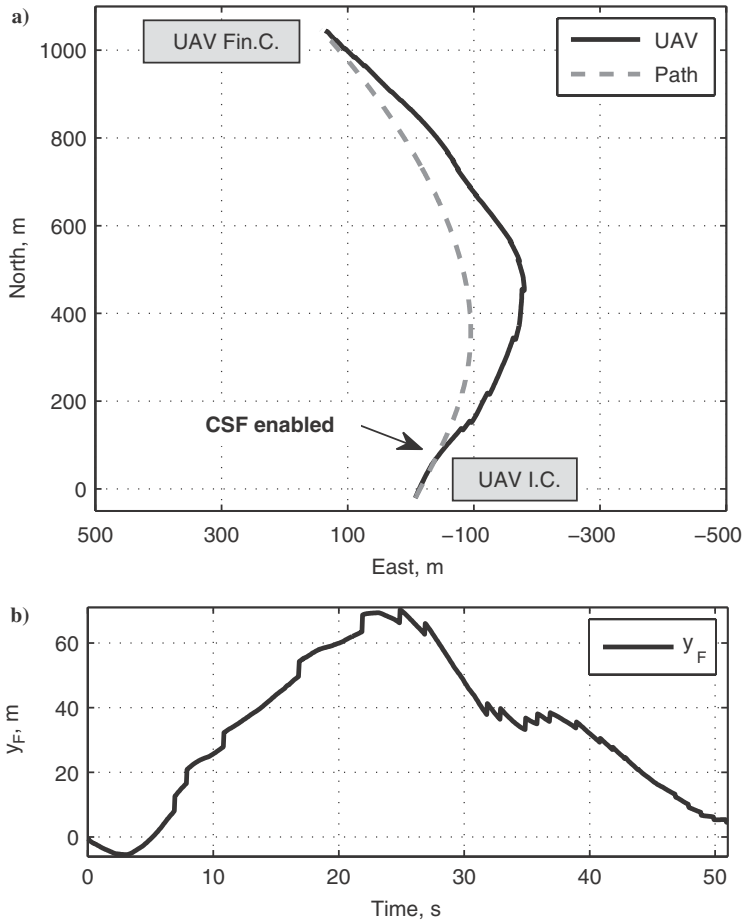


FIG. 7 Ten-deg locked-in-place left-aileron failure: a) two-dimensional projection and b) lateral error.

III. \mathcal{L}_1 ADAPTIVE CONTROL FOR THE NASA AIRSTAR FLIGHT-TEST VEHICLE

During 2007–2010, the NASA Aviation Safety Program created the Integrated Resilient Aircraft Control (IRAC) Project, one of whose primary objectives was to advance and transition adaptive flight control technologies as a means of increasing aviation safety. As part of this project, an \mathcal{L}_1 adaptive control law was flown in a series of remotely piloted flight tests on the NASA AirSTAR

Generic Transport Model (GTM) aircraft, a twin-turbine-powered and dynamically scaled model of a civil transport aircraft, designed to provide a flexible research environment with the ability to conduct rapid prototyping and testing for control algorithms in extremely adverse flight conditions, such as in-flight failure emulation and flight at high-risk upset conditions (see Fig. 8). For further details on the program and the GTM aircraft, see [25, 26].

The \mathcal{L}_1 flight control law was flown at the NASA Wallops Flight Facility, Virginia, in September 2009, and at Fort Pickett, Virginia, in March, June, and September 2010 and in May 2011. The main objective of the flight tests in March and June 2010 was to perform a flight control law evaluation to assess the time-delay margin of the adaptive system, as well as the ability of the adaptive controller to maintain aircraft stability and predictable performance in the presence of aerodynamic stability degradation. Moreover, during the June 2010 deployment, the \mathcal{L}_1 flight control law was tested in poststall flight. The \mathcal{L}_1 control law significantly improved pilot's ability to fly the GTM aircraft at high angles of attack and decreased his workload. During the September 2010 deployment, the \mathcal{L}_1 flight control law was evaluated in a series of offset-to-landing tasks. Moreover, the \mathcal{L}_1 control law was used in support of other research tasks to reduce the pilot's workload and provide tighter acquisition of target flight conditions. One of the research tasks flown during the September 2010 deployment was the calibration of the two air-data vanes placed on each wing tip of the GTM aircraft. In addition, during this same deployment, the \mathcal{L}_1 flight control law was used to support unsteady aerodynamic modeling work at poststall. Finally, during the May 2011 deployment, the \mathcal{L}_1 flight control law continued to support nonlinear unsteady



FIG. 8 NASA AirSTAR flight-test aircraft.

aerodynamic modeling beyond the linear flight regime and also enabled exploration of departure-prone edges of the flight envelope.

This section provides an overview of the results from the flight tests conducted in June and September 2010 and May 2011. The flight tests were performed in strict adherence to the procedures outlined in the flight-test plan [27] and the test cards [28]. For additional details about the design, performance, and flight tests of the \mathcal{L}_1 flight control law for the GTM aircraft, see [14–16].

A. \mathcal{L}_1 ADAPTIVE FLIGHT CONTROL LAW

The \mathcal{L}_1 flight control law developed for the GTM aircraft has as its primary objective achieving tracking for a variety of tasks with guaranteed stability and robustness in the presence of uncertain dynamics, such as changes due to rapidly varying flight conditions during standard maneuvers and unexpected failures. All of these requirements are expected to be achieved while providing level 1 handling qualities [29, 30] under nominal flight conditions, with a graceful degradation under significant adversity.

The \mathcal{L}_1 control law designed for this application consists of a nonadaptive stability augmentation system (SAS) for pitch and roll, and a three-axes angle-of-attack, roll-rate, and angle-of-sideslip adaptive control augmentation system (CAS). The \mathcal{L}_1 adaptive controller thus provides command-tracking capabilities in both nominal and off-nominal conditions as there is no CAS baseline to assist it. The p - β command is one of the standard lateral-directional response types, while the α command was chosen for two reasons. First, modeling research tasks performed on the GTM aircraft require precise α tracking. And second, the AirSTAR operational baseline control law is an α command, and, therefore, the same response type makes a direct control law comparison relevant. Moreover, the adaptive controller provides automatic compensation for the undesirable effects of unmatched uncertainty on the output of the system, which is essential for achieving desired performance, reducing pilot's workload, and improving the aircraft handling qualities. In fact, the effects of unregulated variables, such as air-speed and altitude, unmodeled nonlinearities, cross-coupling between axes, and dynamic asymmetries, may appear as an unmatched component in the inner-loop aircraft dynamics being controlled. For subscale platforms, these dynamics may have timescales comparable to the timescale of the pilot dynamics as well as the dynamics that the pilot is trying to control. Therefore, if the design of the inner-loop flight control system does not account for this uncertainty, then its undesirable effects in the inner-loop dynamics may require excessive pilot compensation or may lead to adverse aircraft-pilot interactions, such as pilot-induced oscillations. In fact, the fast cross-coupling dynamics of the AirSTAR testbed motivated the development of the \mathcal{L}_1 adaptive control architecture with compensation for unmatched uncertainties, which was first presented in [31]. The \mathcal{L}_1 flight control law with its main elements is represented in Fig. 9.

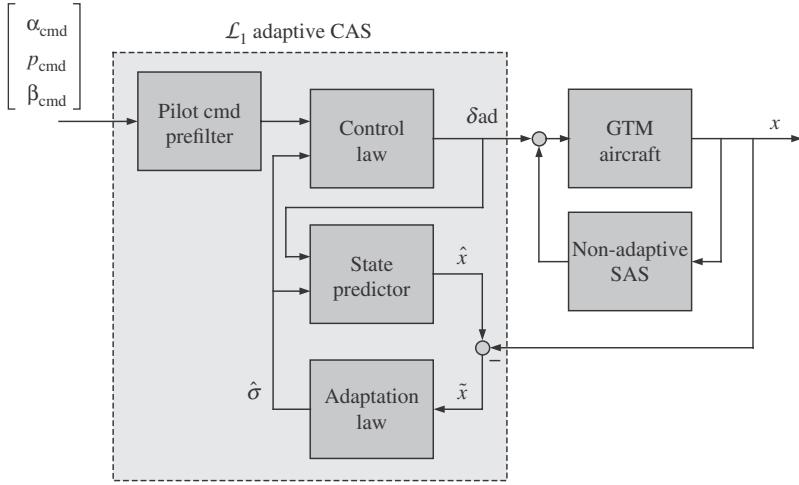


FIG. 9 Block diagram of the \mathcal{L}_1 flight control architecture for the NASA AirSTAR vehicle (simplified).

1. \mathcal{L}_1 ADAPTIVE CONTROL FOR SYSTEMS IN THE PRESENCE OF NONLINEAR UNMATCHED UNCERTAINTIES

The flight control system developed for the AirSTAR flight-test vehicle uses an \mathcal{L}_1 adaptive controller designed for a class of multi-input multi-output uncertain plants in the presence of uncertain system input gain and time- and state-dependent unknown nonlinearities, without enforcing matching conditions [31]. An overview of this \mathcal{L}_1 control architecture is presented next. In particular, the class of systems considered includes general unmatched uncertainty that cannot be addressed by *recursive design methods* developed for strict-feedback systems, semi-strict-feedback systems, pure-feedback systems, and block-strict-feedback systems [32, 33].

Consider the system with unmatched nonlinear uncertainty given by

$$\dot{x}(t) = A_m x(t) + B_m \omega u(t) + f[t, x(t), z(t)] \quad x(0) = x_0 \quad (9a)$$

$$y(t) = Cx(t) \quad (9b)$$

where $x(t) \in \mathbb{R}^n$ is the state vector (measured); $u(t) \in \mathbb{R}^m$ is the control signal; $y(t) \in \mathbb{R}^m$ is the regulated output; $A_m \in \mathbb{R}^{n \times n}$ is a known Hurwitz matrix that defines the desired dynamics for the closed-loop system; $B_m \in \mathbb{R}^{n \times m}$ and $C \in \mathbb{R}^{m \times n}$ are known full-rank constant matrices; $\omega \in \mathbb{R}^{m \times m}$ is the uncertain plant input gain matrix and $f: \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^n$ is an unknown nonlinear function. The initial condition x_0 is assumed to be inside a known set, that is, $\|x_0\|_\infty \leq \rho_0 < \infty$ for some $\rho_0 > 0$. The signal $z(t) \in \mathbb{R}^p$ is the output of the

unmodeled dynamics

$$\dot{x}_z(t) = g[t, x_z(t), x(t)] \quad x_z(0) = x_{z0} \quad (10a)$$

$$z(t) = g_o[t, x_z(t)] \quad (10b)$$

where $x_z(t) \in \mathbb{R}^l$ is the state vector of the internal unmodeled dynamics (*not measurable*); and $g_o: \mathbb{R} \times \mathbb{R}^l \rightarrow \mathbb{R}^p$ and $g: \mathbb{R} \times \mathbb{R}^l \times \mathbb{R}^n \rightarrow \mathbb{R}^l$ are unknown nonlinear functions such that the standard assumptions on existence and uniqueness of solutions are satisfied.

System (9) can be rewritten as

$$\begin{aligned} \dot{x}(t) = & A_m x(t) + B_m \{\omega u(t) + f_1[t, x(t), z(t)]\} \\ & + B_{um} f_2[t, x(t), z(t)] \quad x(0) = x_0 \end{aligned} \quad (11a)$$

$$y(t) = Cx(t) \quad (11b)$$

where $B_{um} \in \mathbb{R}^{n \times (n-m)}$ is a constant matrix such that $B_m^\top B_{um} = 0$ and also $\text{rank}([B_m, B_{um}]) = n$, while $f_1: \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^m$ and $f_2: \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^{(n-m)}$ are unknown nonlinear functions that verify

$$\begin{bmatrix} f_1(t, x(t), z(t)) \\ f_2(t, x(t), z(t)) \end{bmatrix} = B^{-1} f[t, x(t), z(t)], \quad B \triangleq [B_m, B_{um}] \quad (12)$$

In this problem formulation, $f_1(\cdot)$ represents the *matched component* of the uncertainty, whereas $f_2(\cdot)$ represents the *unmatched component*.

Let $X \triangleq [x^\top, z^\top]^\top$, and with a slight abuse of language let $f_i(t, X) \triangleq f_i(t, x, z)$, $i = 1, 2$. It is assumed that for all $t \geq 0$

$$\|f_i(t, 0)\|_\infty \leq B_{i0} \quad i = 1, 2$$

where $B_{i0} > 0$ are known bounds. Next, it is also assumed that, for each $\delta > 0$, there exist positive $K_{1\delta}, K_{2\delta}$ such that

$$\|f_i(t, X_1) - f_i(t, X_2)\|_\infty \leq K_{i\delta} \|X_1 - X_2\|_\infty \quad i = 1, 2$$

for all $\|X_j\|_\infty \leq \delta$, $j = 1, 2$, uniformly in t . Furthermore, assume that there exist $L_z, B_z > 0$ such that, for all $t \geq 0$,

$$\|z_t\|_{\mathcal{L}_\infty} \leq L_z \|x_t\|_{\mathcal{L}_\infty} + B_z$$

The uncertain plant input gain matrix ω is assumed to be an unknown, non-singular, strictly row-diagonally dominant matrix with $\text{sgn}(\omega_{ii})$ known, and also that there exists a known compact convex set Ω such that $\omega \in \Omega \subset \mathbb{R}^{m \times m}$, with $\omega_0 \in \Omega$ being a known nominal plant input gain. Finally, it is also assumed that the transmission zeros of the transfer matrix $H_m(s) = C(sI - A_m)^{-1} B_m$ lie in the open left-half plane.

The control objective is to design an adaptive state-feedback controller to ensure that $y(t)$ tracks the output response of a *desired system* $M(s) \triangleq C(s\mathbb{I} - A_m)^{-1}B_m K_g(s)$ [where $K_g(s)$ is a prefilter] to a given bounded reference signal $r(t)$ *both in transient and steady-state*, while all other signals remain bounded.

Before presenting the adaptive architecture for this class of systems and give an \mathcal{L}_1 -norm stability condition, define

$$\begin{aligned} H_{xm}(s) &\triangleq (s\mathbb{I}_n - A_m)^{-1}B_m \\ H_{xum}(s) &\triangleq (s\mathbb{I}_n - A_m)^{-1}B_{um} \\ H_m(s) &\triangleq CH_{xm}(s) = C(s\mathbb{I}_n - A_m)^{-1}B_m \\ H_{um}(s) &\triangleq CH_{xum}(s) = C(s\mathbb{I}_n - A_m)^{-1}B_{um} \end{aligned}$$

and also define $\rho_{in} \triangleq \|s\mathbb{I} - A_m\|_{\mathcal{L}_1}^{-1} \rho_0$. Furthermore, let $\bar{\gamma}_1$ be an arbitrary small positive constant and, for each $\delta > 0$, define

$$L_{i\delta} \triangleq \frac{\bar{\delta}(\delta)}{\delta} K_{i\bar{\delta}(\delta)}, \quad \bar{\delta}(\delta) \triangleq \max \{ \delta + \bar{\gamma}_1, L_z(\delta + \bar{\gamma}_1) + B_z \} \quad (13)$$

The design of the \mathcal{L}_1 adaptive controller involves a feedback gain matrix $K \in \mathbb{R}^{m \times m}$ and an $m \times m$ strictly proper transfer matrix $D(s)$, which leads, for all $\omega \in \Omega$, to a strictly proper exponentially stable

$$C(s) \triangleq \omega KD(s)[\mathbb{I}_m + \omega KD(s)]^{-1} \quad (14)$$

with dc gain $C(0) = \mathbb{I}_m$. The choice of K and $D(s)$ must ensure that $C(s)H_m^{-1}(s)$ is a proper stable transfer matrix and, for a given ρ_0 , there exists $\rho_r > \rho_{in}$ such that the \mathcal{L}_1 -norm condition

$$\|G_m(s)\|_{\mathcal{L}_1} + \|G_{um}(s)\|_{\mathcal{L}_1} \ell_0 < \frac{\rho_r - \|H_{xm}(s)C(s)K_g(s)\|_{\mathcal{L}_1} \|r\|_{\mathcal{L}_\infty} - \rho_{in}}{L_{1\rho}\rho_r + B_0} \quad (15)$$

is satisfied. In the preceding expression, $G_m(s)$ and $G_{um}(s)$ are defined as

$$\begin{aligned} G_m(s) &\triangleq H_{xm}(s)[\mathbb{I}_m - C(s)] \\ G_{um}(s) &\triangleq [\mathbb{I}_n - H_{xm}(s)C(s)H_m^{-1}(s)C] H_{xum}(s) \end{aligned}$$

while ℓ_0 and B_0 are given by

$$\ell_0 \triangleq \frac{L_{2\rho_r}}{L_{1\rho_r}}, \quad B_0 \triangleq \max \left\{ B_{10}, \frac{B_{20}}{\ell_0} \right\}$$

The elements of the \mathcal{L}_1 adaptive control architecture are introduced next. The *state predictor* is given by

$$\dot{\hat{x}}(t) = A_m \hat{x}(t) + B_m [\omega_0 u(t) + \hat{\sigma}_1(t)] + B_{um} \hat{\sigma}_2(t) \quad \hat{x}(0) = x_0 \quad (16a)$$

$$\hat{y}(t) = C \hat{x}(t) \quad (16b)$$

where $\hat{\sigma}_1(t) \in \mathbb{R}^m$ and $\hat{\sigma}_2(t) \in \mathbb{R}^{n-m}$ are the adaptive estimates of the matched and unmatched uncertainties, respectively. These estimates are driven by piecewise-constant *adaptation laws*

$$\begin{bmatrix} \hat{\sigma}_1(t) \\ \hat{\sigma}_2(t) \end{bmatrix} = - \begin{bmatrix} \mathbb{I}_m & 0 \\ 0 & \mathbb{I}_{n-m} \end{bmatrix} B^{-1} \Phi^{-1}(T_s) \mu(T_s) \tilde{x}(iT_s) \quad t \in [iT_s, (i+1)T_s] \quad (17)$$

for $i = 0, 1, 2, \dots$, where $T_s > 0$ is the *adaptation sampling time*, $\tilde{x}(t) \triangleq \hat{x}(t) - x(t)$ is the prediction error, B was introduced in Eq. (12), and

$$\Phi(T_s) \triangleq A_m^{-1} [\exp(A_m T_s) - \mathbb{I}], \quad \mu(T_s) = e^{A_m T_s}$$

The *control law* is given by

$$u(s) = -KD(s)\hat{\eta}(s) \quad (18)$$

where $\hat{\eta}(s)$ is the Laplace transform of the signal

$$\hat{\eta}(t) \triangleq \omega_0 u(t) + \hat{\eta}_1(t) + \hat{\eta}_{2m}(t) - r_g(t) \quad (19)$$

with $r_g(s) \triangleq K_g(s)r(s)$, $\hat{\eta}_{2m}(s) \triangleq H_m^{-1}(s)H_{um}(s)\hat{\eta}_2(s)$, where $\hat{\eta}_2(t) \triangleq \hat{\sigma}_2(t)$ and $\hat{\eta}_1(t) \triangleq \hat{\sigma}_1(t)$. The term $\hat{\eta}_1(t)$ intends to compensate for the matched part of the uncertainties, while the term $\hat{\eta}_{2m}(t)$ is meant to compensate for the undesirable effects of the unmatched uncertainties at the output of the system.

Notice that conventional design methods from multivariable control theory can be used to design the prefilter $K_g(s)$ to achieve desired decoupling properties. As an example, assuming the matrix $CA_m^{-1}B_m$ is invertible, if one chooses $K_g(s)$ as the constant matrix $K_g = -(CA_m^{-1}B_m)^{-1}$, the diagonal entries of the desired transfer matrix $M(s) = C(s\mathbb{I}_n - A_m)^{-1}B_m K_g$ have dc gain equal to one, while the off-diagonal entries have zero dc gain.

The \mathcal{L}_1 reference system for this architecture is given by

$$\dot{x}_{\text{ref}}(t) = A_m x_{\text{ref}}(t) + B_m \{\omega u_{\text{ref}}(t) + f_1[t, x_{\text{ref}}(t), z(t)]\} + B_{um} f_2[t, x_{\text{ref}}(t), z(t)] \quad x_{\text{ref}}(0) = x_0 \quad (20a)$$

$$u_{\text{ref}}(s) = -\omega^{-1}C(s)[\eta_{1\text{ref}}(s) + H_m^{-1}(s)H_{um}(s)\eta_{2\text{ref}}(s) - K_g(s)r(s)] \quad (20b)$$

$$y_{\text{ref}}(t) = Cx_{\text{ref}}(t) \quad (20c)$$

where $\eta_{i\text{ref}}(s)$ is the Laplace transform of $\eta_{i\text{ref}}(t) \triangleq f_i[t, x_{\text{ref}}(t), z(t)]$, $i = 1, 2$.

The next theorem summarizes the main result for this adaptive architecture.

Theorem 2 ([31] Theorem 1)

Let the feedback gain K and the filter $D(s)$ be chosen to satisfy the \mathcal{L}_1 -norm condition in Eq. (15). Then, the reference system in Eq. (20) is bounded-input, bounded-output stable. Moreover, there exists an adaptation sampling time T_s such that

$$\|\tilde{x}\|_{\mathcal{L}_\infty} \leq \gamma_0(T_s) \quad (21)$$

$$\|x_{\text{ref}} - x\|_{\mathcal{L}_\infty} \leq \gamma_1(T_s) \quad (22)$$

$$\|u_{\text{ref}} - u\|_{\mathcal{L}_\infty} \leq \gamma_2(T_s) \quad (23)$$

where $\gamma_0(T_s)$, $\gamma_1(T_s)$, and $\gamma_2(T_s)$ are known class \mathcal{K} functions of T_s .

Notice that the functions $\gamma_0(T_s)$, $\gamma_1(T_s)$, and $\gamma_2(T_s)$ can be constructed based on knowledge of the uncertainty bounds and the adaptive control system parameters. For details see [31] and Chap. 3.3 in [7].

Thus, the tracking error between $y(t)$ and $y_{\text{ref}}(t)$, as well as $u(t)$ and $u_{\text{ref}}(t)$, is uniformly bounded by a constant that can be arbitrarily reduced by decreasing the adaptation sampling time T_s . Therefore, arbitrarily close tracking performance can be achieved, both in transient and steady state, for both system's input and output signals simultaneously by adapting sufficiently fast.

2. TUNING

The \mathcal{L}_1 adaptive flight control law was designed to satisfy the same requirements as any standard classical flight control system, namely, good flying qualities, precision tracking, reduced control-surface activity, and graceful degradation in the presence of faults and failures. Specifically, the design of the \mathcal{L}_1 control law for the GTM is based on the linearized dynamics of the aircraft at a nominal flight condition corresponding to an equivalent airspeed of 80 kt and an altitude of 1000 ft. For design purposes, these linear dynamics were further simplified to include only short-period dynamics for the longitudinal axis, roll rate, sideslip angle, and yaw rate for the lateral-directional dynamics. Because the airplane is level 1 at this flight condition, the nominal desired dynamics of the linear state predictor were chosen to be similar to the linearized dynamics of the airplane. Some additional damping was added to both longitudinal and directional dynamics of the state predictor, while the lateral dynamics were set to be faster than the original dynamics to satisfy performance specifications. To improve the handling qualities of the airplane, a linear prefilter was added to the adaptive flight control law to ensure desired decoupling properties as well as desired command-tracking performance. Overdamped second-order low-pass filters with unity dc gain were used in all control channels, while their bandwidths were set to ensure a time-delay margin of at least 0.125 s and a gain margin of at least 6 dB. Finally, the adaptation sampling time was set to 1/600 s,

which corresponds to the execution speed of the AirSTAR flight control computer. Recall that \mathcal{L}_1 adaptive controller takes full advantage of the available computation speeds to improve the performance bounds. The same control parameters for the prefilter, low-pass filters, and adaptation rate are used across the entire flight envelope, including the poststall angle-of-attack region, with no scheduling or reconfiguration.

B. FLIGHT CONTROL LAW EVALUATION

The formal evaluation of the flight control laws on the AirSTAR GTM consists of a series of piloted tasks that try to assess their robustness and performance in both nominal and off-nominal flight conditions. On one hand, robustness of the flight control laws is measured in terms of the time-delay margin, which is determined in flight by progressively increasing the latency in the control channel until instability occurs. On the other hand, performance of the control laws is evaluated through a series of tasks designed to test the loss-of-control prevention and recovery capabilities of the closed-loop system. In particular, these tasks include flight under aerodynamic stability degradation, poststall flight and recovery, and handling qualities assessment based on Cooper – Harper ratings (CHR) as well as subjective pilot comments. A detailed description of these evaluation tasks can be found in [34]. A brief overview of representative results from the flights under aerodynamic stability degradation and the offset-to-landing tasks are presented next. References [8] and [16] provide additional data and analysis of these results, while detailed descriptions of the latency-injection task and the poststall and recovery evaluation tasks can be found in [15] with the corresponding flight-test data.

1. FLIGHT UNDER STABILITY DEGRADATION

In the stability degradation tasks, both the longitudinal and lateral stability of the aircraft are simultaneously degraded, while reducing pitch-control authority by 50%. The stability degradation task is introduced gradually; starting with a 50% degradation, the stability is reduced in decrements of 25% until the closed-loop system with the tested flight controller is unstable. For this task, the pilot is asked to fly the aircraft in a figure-eight pattern at approximately 900 ft and 80 kt, while pitch- and roll-doublet wavetrains are superimposed on the pilot's stick commands during the straight legs of the pattern to excite the system dynamics. The \mathcal{L}_1 flight control law tested in June 2010 was able to maintain aircraft stability for up to 100% simultaneous degradation in longitudinal and lateral aerodynamic stability with an additional 50% reduction in pitch-control authority.

To provide a basis for comparison, the nominal response of the aircraft with the \mathcal{L}_1 flight control law, which is considered to have solid level 1 handling qualities according to pilot ratings and comments, is presented in Figs. 10 and 11. In particular, Fig. 10 shows the pitch-axis wavetrain with the corresponding lateral-directional response, while Fig. 11 presents the roll-axis wavetrain with the

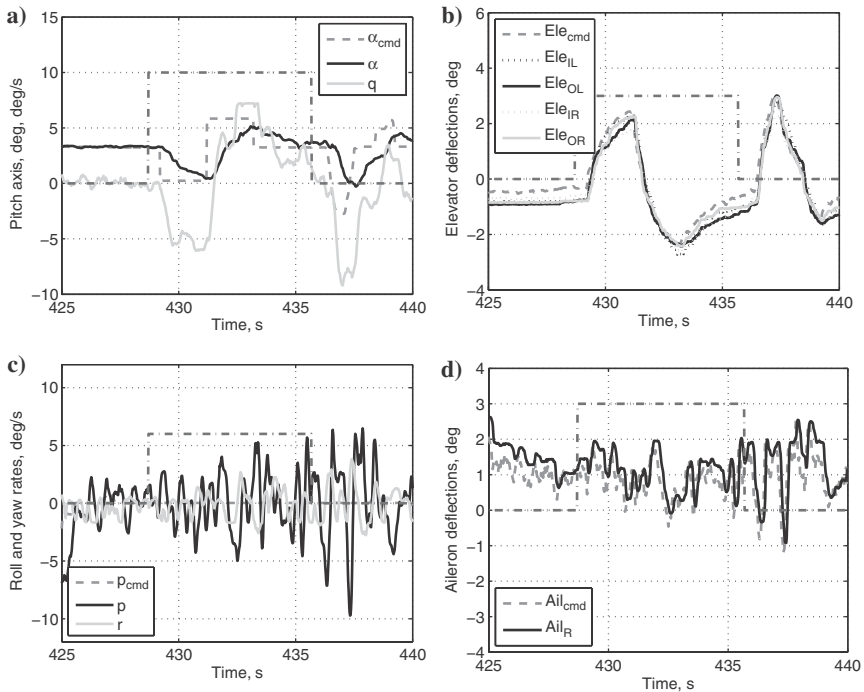


FIG. 10 Pitch-doublet response for the nominal aircraft (The injection of the wavetrain is indicated in the plots with a dark-grey dashed-dotted line.): a) pitch-axis response, b) elevator deflections, c) lateral-directional response, and d) right-aileron deflection.

corresponding longitudinal response. The flight control law, which provides a first-order-like response for both channels, is able to isolate the commanded response to the appropriate axis, ensuring reduced cross-coupling between channels. During the injection of the pitch-axis wavetrain, for example, the roll and yaw rates do not exceed ± 5 deg/s, which is considered to be within the turbulence level; see Fig. 10c. Similarly, during the injection of the roll-axis wavetrain, both the angle of attack and sideslip angle are maintained within ± 0.5 deg of the commanded values; see Figs. 11c and 11e. The main comment from the pilot during the postflight debrief is that the \mathcal{L}_1 flight control law provides “a nice Level 1 flying airplane for light turbulence conditions.”

Next, the performance of the closed-loop adaptive system in the presence of aerodynamic stability degradation is analyzed. Recall that the stability degradation is applied simultaneously in the longitudinal and lateral axes while the elevator effectiveness is reduced by 50%. The reduction in longitudinal stability is obtained by closing a destabilizing feedback loop from angle of attack to inboard elevators, while the spoilers are used asymmetrically to reduce lateral stability through roll

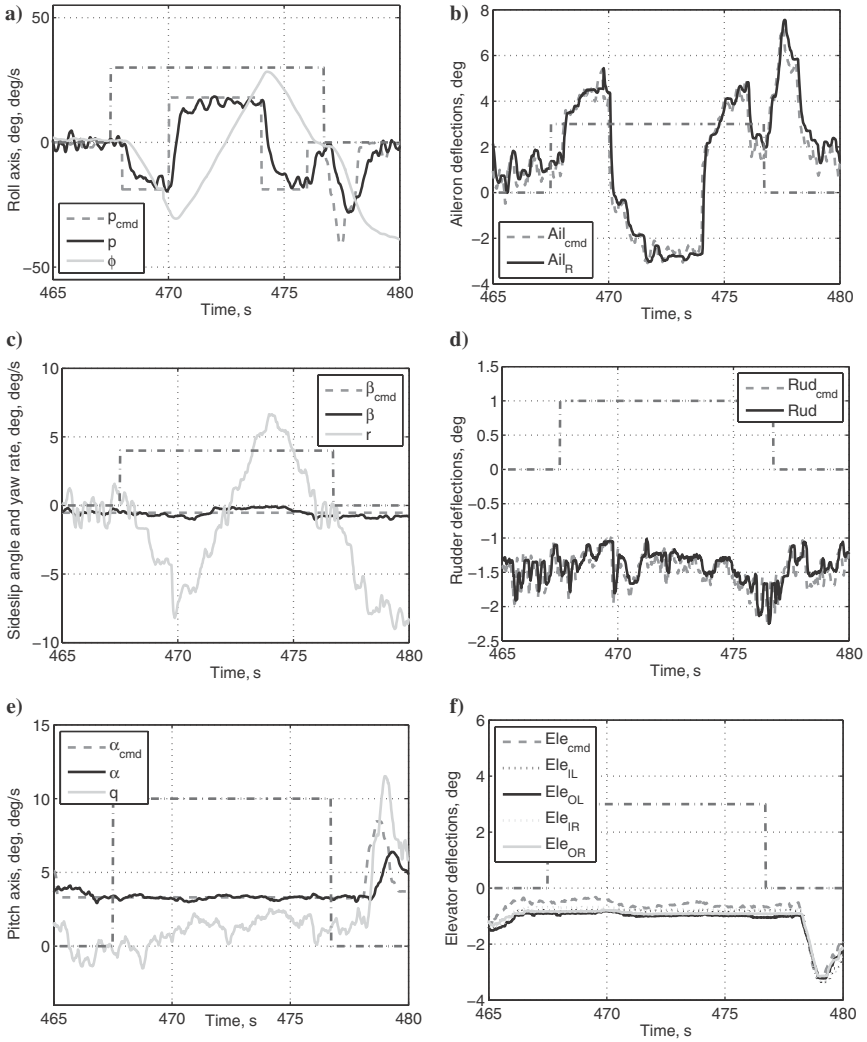


FIG. 11 Roll-doublet response for the nominal aircraft (The injection of the wavetrain is indicated in the plots with a dark-grey dashed-dotted line.): a) roll-axis response, b) right-aileron deflection, c) directional response, d) rudder deflection, e) pitch-axis response, and f) elevator deflections.

damping. The results for the case of 50% stability degradation are similar to the results for the nominal aircraft shown in Figs. 10 and 11. The main difference is a higher control-surface activity to compensate for turbulence, especially in the roll channel. The pilot, however, reported that he did not detect a perceptible

difference from flying the nominal aircraft. The results for the 50% degradation case can be found in [15]. The case of 75% stability degradation, which is considered to represent a severe failure, is illustrated in Figs. 12 and 13. As shown in the figures, the closed-loop system experiences a slight degradation in roll-rate tracking performance, while the pitch-axis response remains similar to the response of the nominal aircraft. In addition, there is more aileron activity to compensate for the turbulence-induced roll rate.

For the case of 100% stability degradation, the \mathcal{L}_1 flight control law was not able to maintain prescribed performance; see Figs. 14 and 15. The roll response exhibits oscillatory behavior as well as increased aileron activity, resulting in an evident deterioration in the closed-loop system performance. In particular, Fig. 14 shows that the pitch-axis closed-loop damping is reduced, as evidenced by overshoot in the angle-of-attack response and larger amplitudes in pitch rate. Moreover, as shown in Fig. 14c, the pitch-axis doublet induces ± 15 -deg/s

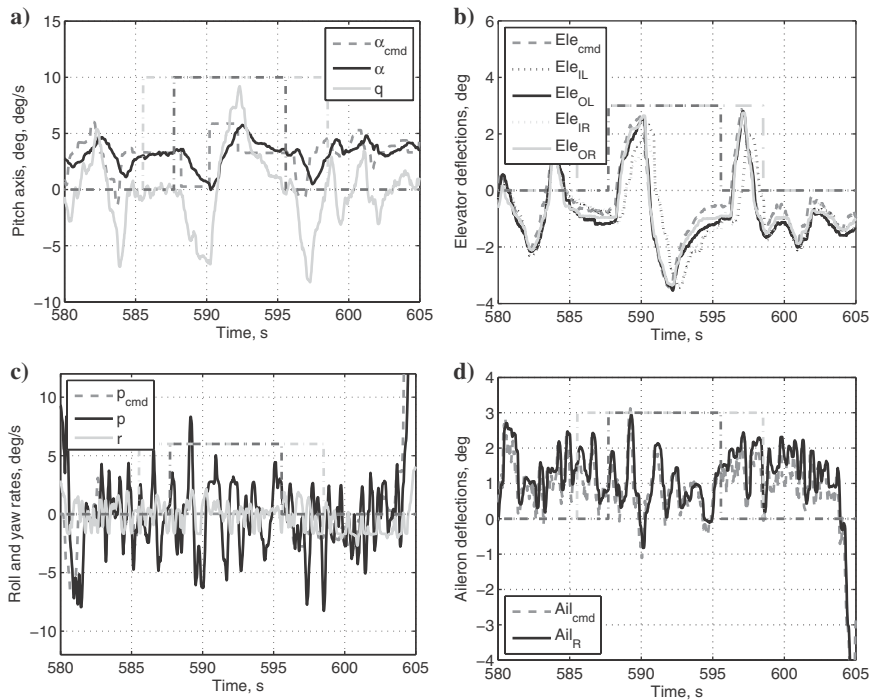


FIG. 12 Pitch-doublet response for the case of 75% stability degradation (The injection of the wavetrain is indicated in the plots with a dark-grey dashed-dotted line, while the injection of the stability degradation is indicated with a light-grey dashed-dotted line.): a) pitch-axis response, b) elevator deflections, c) lateral-directional response, and d) right-aileron deflection.

oscillations in roll rate. The pilot observed roll ratcheting during the injection of the pitch-axis wavetrain. In Fig. 14b, the use of the inboard elevators to produce the 100% longitudinal stability degradation is evident; in fact, as already

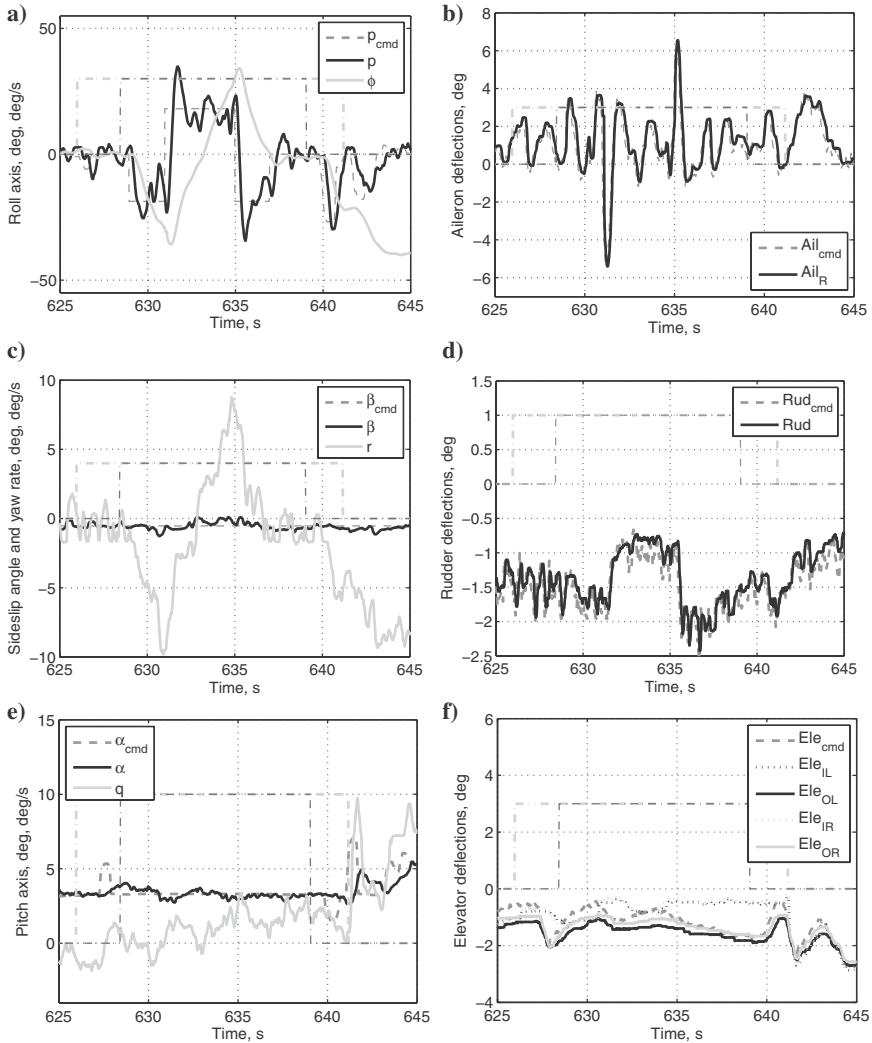


FIG. 13 Roll-doublet response for the case of 75% stability degradation (The injection of the wavetrain is indicated in the plots with a dark-grey dashed-dotted line, while the injection of the stability degradation is indicated with a light-grey dashed-dotted line.): a) roll-axis response, b) right-aileron deflection, c) directional response, d) rudder deflection, e) pitch-axis response, and f) elevator deflections.

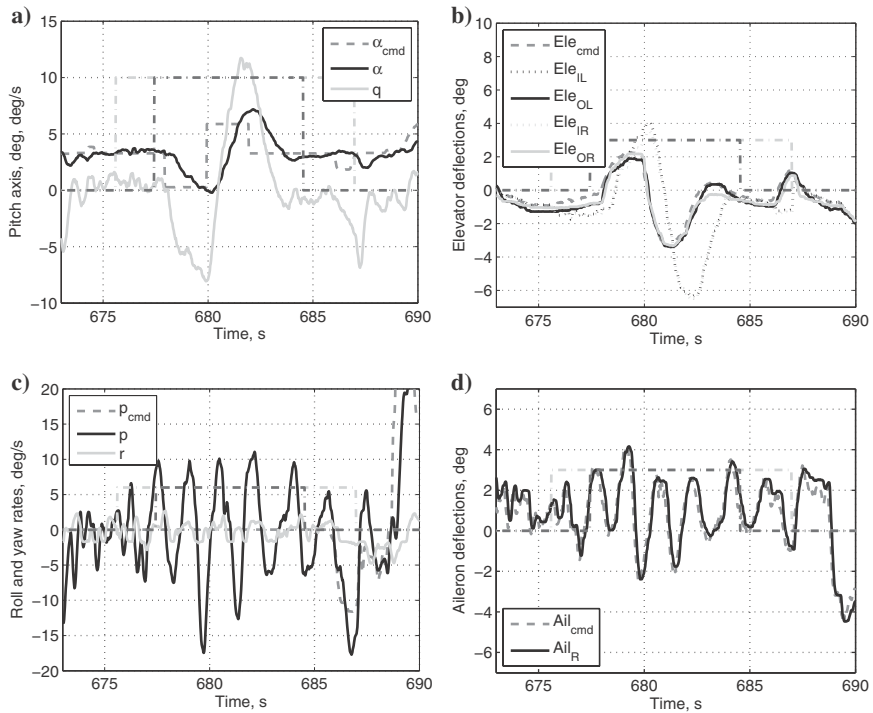


FIG. 14 Pitch-doublet response for the case of 100% stability degradation (The injection of the wavetrain is indicated in the plots with a dark-grey dashed-dotted line, while the injection of the stability degradation is indicated with a light-grey dashed-dotted line.): a) pitch-axis response, b) elevator deflections, c) lateral-directional response, and d) right-aileron deflection.

mentioned, the inboard elevators are scheduled with angle of attack to create a destabilizing feedback loop and thus do not follow the elevator command generated by the \mathcal{L}_1 flight control law. During the injection of the roll-axis wavetrain, the closed-loop system exhibits a large overshoot and ± 20 -deg/s sustained oscillations in roll rate, which seems to indicate that the controller is reaching the limit of its ability to provide robust stability in the roll channel; see Fig. 15. Also, despite the controller performance degradation in the roll axis, the angle-of-attack and sideslip-angle tracking errors are maintained within ± 1 deg.

During the June 2010 deployment, the \mathcal{L}_1 flight control law was also flown under 125% stability degradation. Flight-test results for this case are shown in Fig. 16. While the closed-loop pitch response was still stable (Fig. 16a), the roll channel presented divergent oscillations induced by the pitch-doublet wavetrain (Fig. 16c). At approximately 760 s, the pilot called the “knock it off,” a signal for abandoning a particular maneuver/fault, and the stability degradation fault

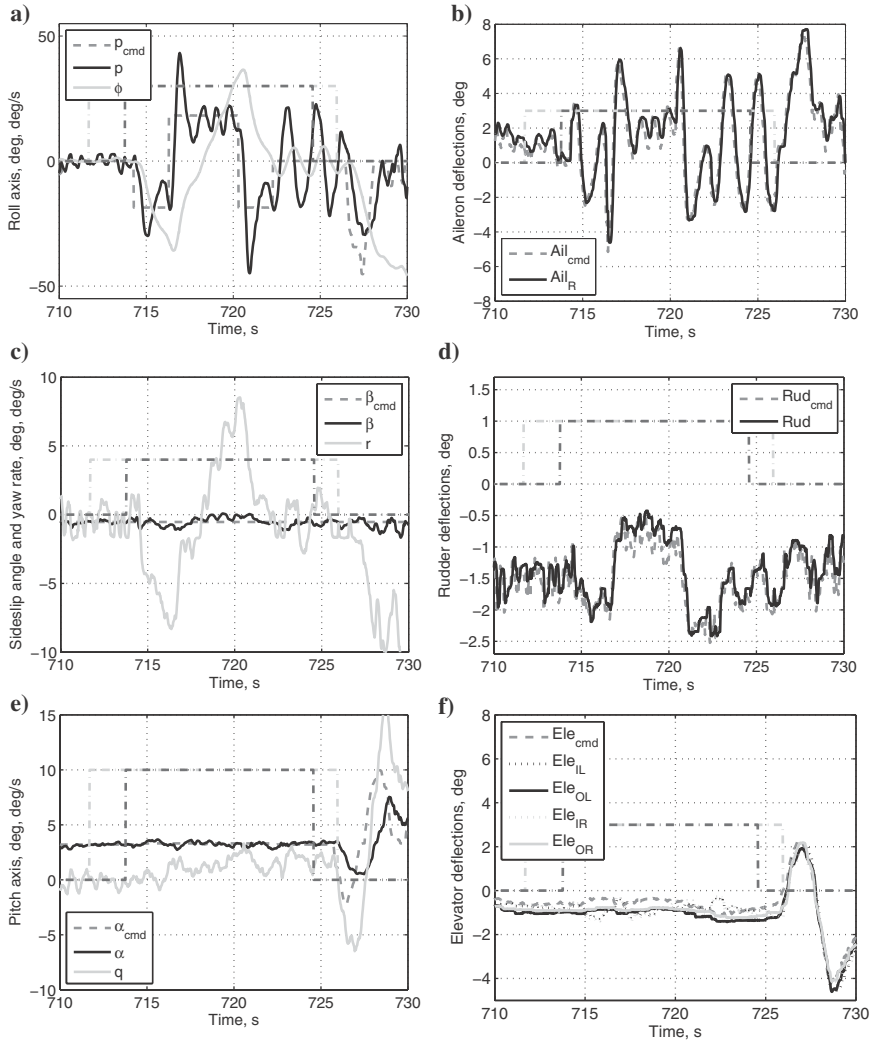


FIG. 15 Roll-doublet response for the case of 100% stability degradation (The injection of the wavetrain is indicated in the plots with a dark-grey dashed-dotted line, while the injection of the stability degradation is indicated with a light-grey dashed-dotted line.): a) roll-axis response, b) right-aileron deflection, c) directional response, d) rudder deflection, e) pitch-axis response, and f) elevator deflections.

was immediately disengaged. As soon as the fault was disengaged, the controller recovered its nominal performance and started tracking pilot's commands. Fast roll recovery is particularly evident in Fig. 16c.

2. OFFSET-TO-LANDING TASKS

To conduct a handling-qualities evaluation of the research control laws, a simulated offset-to-landing task was developed as a high workload task. Offset-to-landing is considered a challenging maneuver due to proximity to the ground, requirement for high level of precision, and associated workload. The maneuver consists of an “s-turn” as the pilot attempts to realign the aircraft with the center of the runway during the final part of the approach to land. The pilot’s task is to penetrate a touchdown zone target box at a 3-deg descent angle with the wings level. This maneuver serves to severely tax the aircraft-pilot system in the roll and pitch channels in order to bring out any unwanted handling-qualities deficiencies in the system.

For the AirSTAR GTM, the initial task condition is a final approach at 80 kt characterized by an offset from the centerline of the virtual runway that is 100 ft to

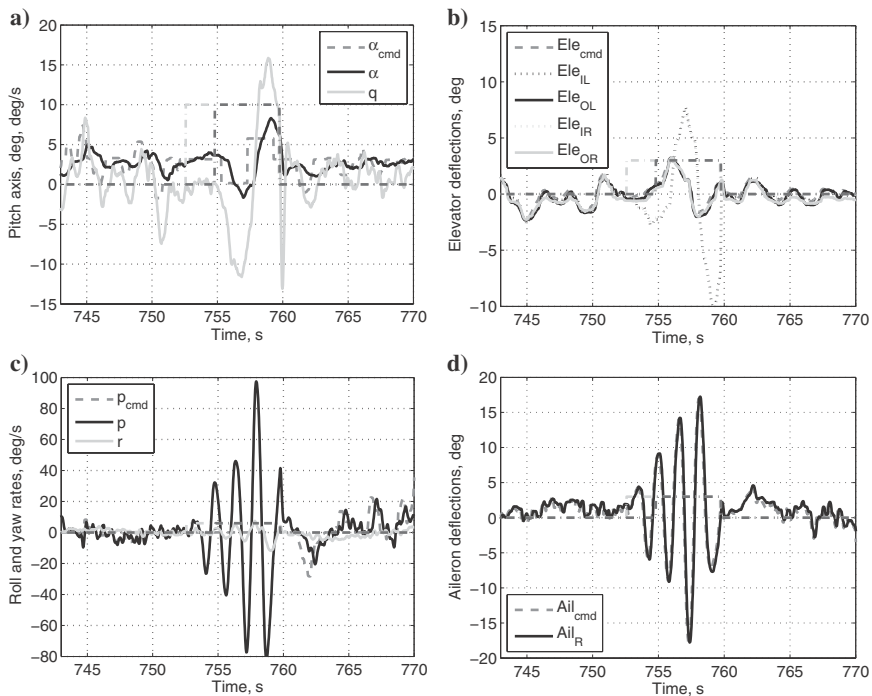


FIG. 16 Pitch-doublet response for the case of 125% stability degradation (The injection of the wavetrain is indicated in the plots with a dark-grey dashed-dotted line, while the injection of the stability degradation is indicated with a light-grey dashed-dotted line.):
 a) pitch-axis response, b) elevator deflections, c) lateral-directional response, and d) right-aileron deflection.

the left of the centerline, 100 ft above the runway, and 1800 ft downrange from a desired touchdown box marked on the runway. The task requirements are set on aircraft attitude, flight-path angle, and touchdown position on the runway, with the following performance criteria:

1. Desired performance:
 - (a) 164- × 12-ft touchdown zone
 - (b) $|\phi| < 10$ deg
 - (c) $|\Delta\gamma| < 1$ deg
2. Adequate performance:
 - (a) 363- × 24-ft touchdown zone
 - (b) $|\phi| < 20$ deg
 - (c) $|\Delta\gamma| < 3$ deg

Data used to assess the effectiveness of the flight control laws included time-history parameter recordings, pilot comments, and a pilot-assigned Cooper–Harper rating scale.

This particular offset-to-landing task with the specified performance criteria is considered to be demanding for a pilot under nominal aircraft conditions. To evaluate the performance of the flight control laws, the task is performed also for the cases of 100 and 125% stability degradation in both longitudinal and lateral axes with an additional 50% reduction in pitch-control authority.

The offset-to-landing task evaluation with the \mathcal{L}_1 flight control law was conducted during the September 2010 deployment. To place the results in an appropriate context, it is convenient to refer to the results reported in [34] and note that the GTM aircraft with direct stick-to-surface control (no feedback control law) is rated CHR 4 (border line CHR 5), level 2 flying qualities, for the offset-to-landing task of the nominal aircraft. For the neutral stability case, 100% roll- and pitch-axis stability degradation, the pilot rated the task as CHR 10, that is, aircraft uncontrollable with catastrophic landing expected. The offset-to-landing task for the \mathcal{L}_1 control law was flown under light turbulence conditions, which typically exacerbates stability degradation especially in roll axis because the destabilizing feedback loop responds to turbulence-induced roll rate. Hence, the implication of turbulence is that the control surface deflections required to make aircraft neutrally stable, 100% stability degradation, would, in reality, be larger and make the aircraft slightly unstable. Under nominal aircraft stability, the \mathcal{L}_1 control law is rated CHR 3, level 1 handling qualities. For the case of 100% stability degradation, the \mathcal{L}_1 control law experiences some handling-qualities degradation and is rated CHR 5, level 2. Finally, for the case of 125% stability degradation, the pilot achieves a “safe landing” with CHR 7, level 3 handling qualities. Table 1 provides the concise summary of comments and CHRs that the AirSTAR research pilot provided for the offset-to-landing task evaluation with the \mathcal{L}_1 flight control law. The reader is referred to [16] for the flight-test data of these offset-to-landing tasks.

TABLE 1 SUMMARY OF COMMENTS AND COOPER–HARPER RATINGS FROM THE OFFSET-TO-LANDING TASK (FROM [34])

Flight No.	Turbulence	Stability Degradation	Controllable?	Performance	Workload Tolerable?	Pilot Compensation	CHR
41	Light	Nominal	Yes	Desired	Yes	Minimal	3
		100%	Yes	Adequate	Yes	Considerable	5
		125%	Yes	Not adequate	Yes	Max tolerable for performance	7

C. \mathcal{L}_1 CONTROL IN SUPPORT OF LARGE ENVELOPE MODELING TASKS

From the perspective of the Aviation Safety Program Project, interest in stall and poststall flight is two-fold. First, there is an interest in exploiting the AirSTAR facility as a flying wind tunnel to model unsteady nonlinear aerodynamics encountered in near- and poststall flight regimes. Second, high angle-of-attack flight provides a challenging scenario for adaptive controllers in which the dynamics of the aircraft experience significant changes and are characterized by uncertain nonlinear phenomena. The primary challenges associated with poststall angle of attack are rapidly changing roll-damping characteristics and an abrupt wing drop due to asymmetric wing stall. Roll asymmetry is expected to occur at an angle of attack of about 13 deg. Also, roll damping characteristics are expected to rapidly change from stable to unstable in the 10–12 deg angle-of-attack range. At angles of attack above stall, roll damping is expected to improve and aerodynamic asymmetry is expected to decrease. The defining stall characteristic is expected to be an unstable pitch break occurring at an angle of attack of 13 deg; moreover, at poststall 15-deg angle of attack, static and dynamic longitudinal stability is expected to be low.

By the September 2010 deployment, the \mathcal{L}_1 flight control law had established itself as a reliable and predictable tool to be used in support of other research tasks in order to reduce research pilot's workload and provide tighter acquisition of target flight conditions. Part of the September 2010 deployment and all of the May 2011 deployment dealt with instrument calibration and large flight envelope modeling tasks. The research tasks supported by the \mathcal{L}_1 adaptive control law are outlined in Table 2. A brief overview of representative results from these flights is presented next.

TABLE 2 SUMMARY OF RESEARCH TASKS SUPPORTED BY THE \mathcal{L}_1 ADAPTIVE CONTROL LAW

Research Task	Subtask	Deployment	Flight No.
Air-data vane calibration	α vane calibration	September 2010, May 2011	28, 56
	β vane calibration	September 2010, May 2011	29, 31, 56
Unsteady aerodynamic modeling work	Poststall α tracking	September 2010	31, 35, 52
	Roll forced oscillations	May 2011	49, 50, 53, 56, 57
Exploration of departure-prone edges	α sweep from low angles, through stall, to departure	May 2011	54, 55, 58

1. AIR-DATA VANE CALIBRATION

One of the research tasks flown during the September 2010 deployment was the calibration of the two air-data vanes placed on each wing tip of the GTM aircraft (see Fig. 8), the angle-of-attack and the angle-of-sideslip measurements of which are used for modeling and control. In these calibration tasks, the \mathcal{L}_1 flight control law was employed to provide precision tracking, execution within tighter limits, and reduction of pilot workload.

Standard methods for angle-of-attack and angle-of-sideslip vane calibration can be found in [35]. For the angle-of-attack vane calibration, the pilot sets a predetermined thrust level (specified in terms of percent rpm) and adjusts the longitudinal stick to command a desired angle-of-attack response. Two different strategies are used in these calibration tasks, a “variable α strategy” and a “constant α strategy.” The set of maneuvers for angle-of-attack vane calibration is flown manually by the research pilot. On the other hand, the angle-of-sideslip vane calibration involves flat turns with a sideslip-angle ramp command to various steady-state values, which are then held until the research range boundary is approached. The sideslip-angle command is a generated wavetrain, while the pilot is responsible for flying the other axes.

Angle-of-attack Vane Calibration

As just mentioned, the angle-of-attack vane calibration was approached with two different strategies. In the first one, referred to as “variable α strategy” and illustrated in Fig. 17, the pilot sets a specific throttle level and pulls the stick to force the angle of attack to ramp up steadily. The throttle setting, pilot stick input, and the subsequent angle-of-attack response are illustrated in Figs. 17a–17c. The maneuver is repeated twice, from about 872 to 892 s and from 920 to 940 s, with stall recovery and turn in the interval 892–920 s.

In the second strategy, referred to as “constant α strategy” and illustrated in Fig. 18, the pilot sets a specific throttle level and commands a constant angle of attack. The particular values of angle of attack used in calibration were 5, 8, 10, and 12 deg. The precision of the angle-of-attack tracking near stall and in stall is illustrated respectively in Figs. 19a and 20a. In addition to provide improved angle-of-attack tracking, the \mathcal{L}_1 control law stabilizes the roll axis by compensating for the rapid change of the roll dynamics in this angle-of-attack region, see Figs. 19c and 20c.

Angle-of-Sideslip Vane Calibration

The angle-of-sideslip vane calibration involved flat-turn maneuvers with angle-of-sideslip ramp commands for various steady-state values, with a maximum of $|\beta_{\text{cmd}}| = 8$ deg dictated by maximum aileron deflection to counteract roll rate and maintain the flat turn. The sideslip-angle command was ramped up at 2 deg/s and at 1 deg/s and then held at angles of sideslip of ± 2 , ± 4 , ± 6 , and ± 8 deg. This set of maneuvers required tight tracking of the sideslip-angle

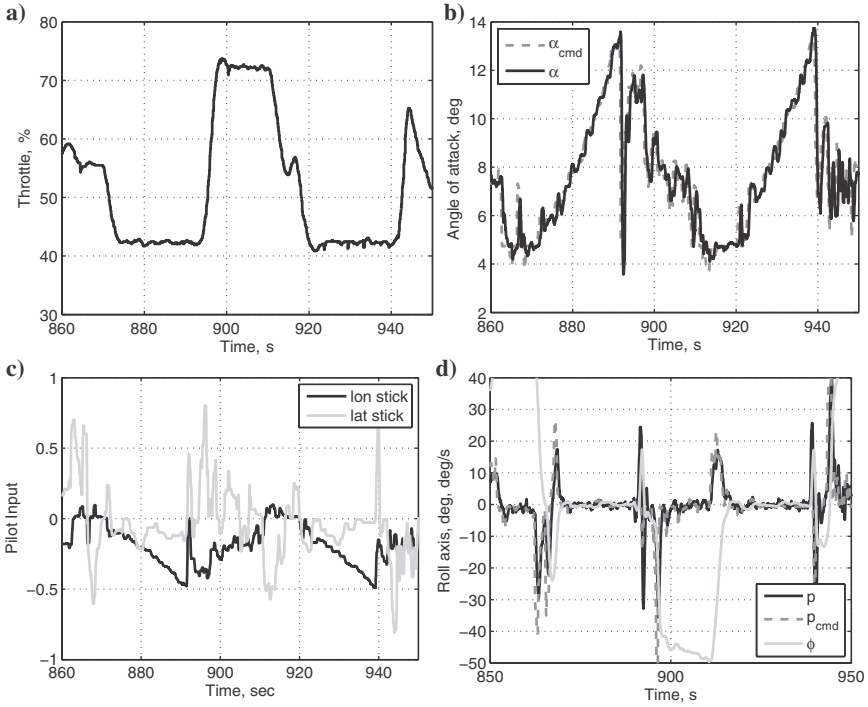


FIG. 17 Angle-of-attack vane calibration using variable α strategy: a) commanded throttle setting, b) angle-of-attack response, c) pilot stick input, and d) lateral response.

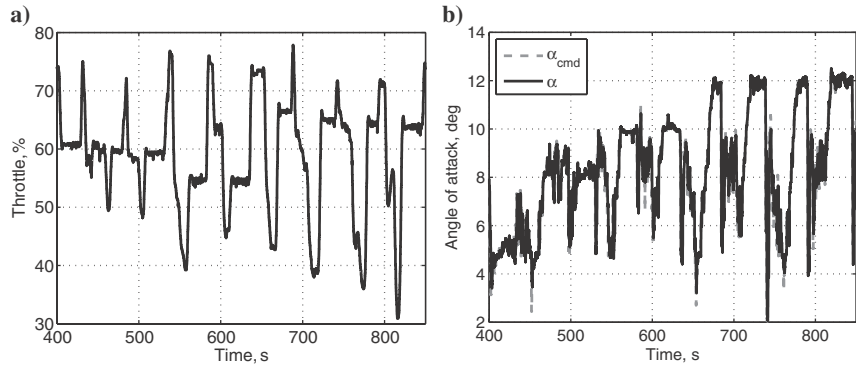


FIG. 18 Angle-of-attack vane calibration using constant α strategy: a) commanded throttle setting and b) angle-of-attack response.

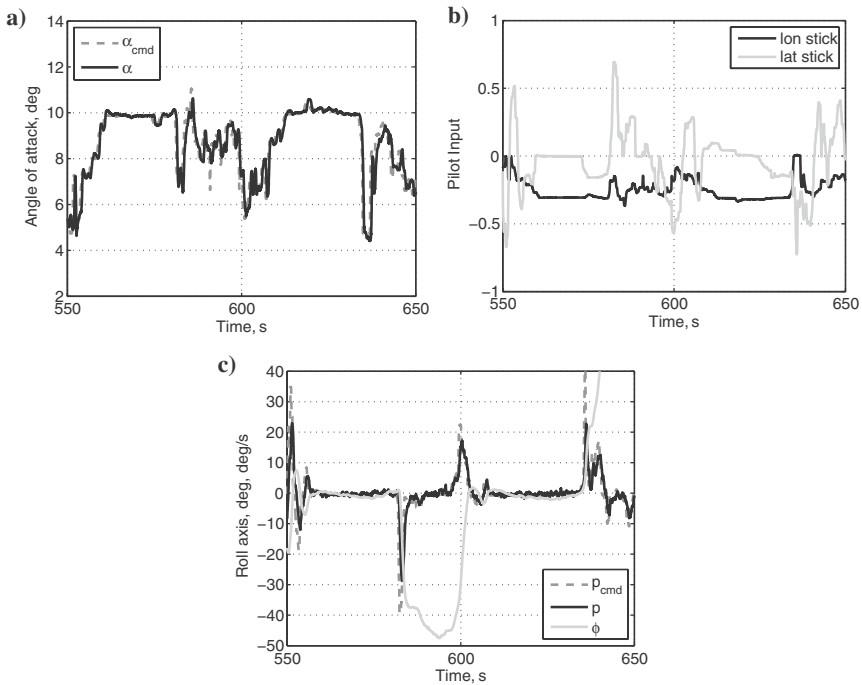


FIG. 19 Angle-of-attack vane calibration using constant α strategy, 10-deg angle-of-attack acquisition: a) angle-of-attack response, b) pilot stick input, and c) lateral response.

command and minimal roll dynamics (desired bank angle is less than ± 2 deg; adequate, less than ± 4 deg). The sideslip-angle command is provided by an automated wavetrain, while the pilot flies the roll and pitch axes. An entire flight was dedicated to angle-of-sideslip vane calibration, which is illustrated in Fig. 21. Note that each value for commanded angle of sideslip was flown twice, on the up- and downwind legs of the figure-eight pattern.

Precision tracking of the $+8$ and -8 deg sideslip-angle commands, together with the associated dynamics, are shown in Figs. 22 and 23. The flat turn at $+8$ deg of angle of sideslip with the corrective pilot stick inputs to maintain the flat turn are shown in Fig. 22. Interestingly, the results show that the maximum aileron deflection in the GTM is not enough to maintain a flat turn at $+8$ deg at the research reference airspeed. In fact, during the two flat turns, the ailerons were completely saturated (Fig. 22e). In spite of this fact, the pilot intuitively commanded lateral stick in order to counteract the tendency of the aircraft to roll during the flat turn. Because the design of the \mathcal{L}_1 control law for the lateral and directional channels is integrated, lateral stick also contributes to the rudder command. The result of the pilot's action, thus, was that the commanded

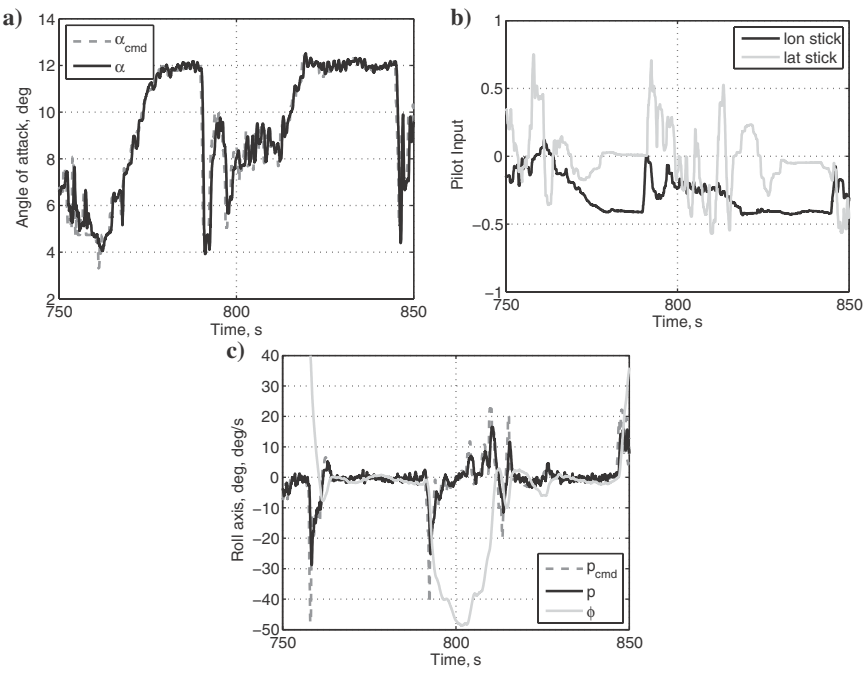


FIG. 20 Angle-of-attack vane calibration using constant α strategy, 12-deg angle-of-attack acquisition: a) angle-of-attack response, b) pilot stick input, and c) lateral response.

lateral stick reduced the rudder command generated by the +8-deg sideslip-angle command, resulting in a steady flat turn around 7.5 deg of sideslip angle with an actual roll rate of less than ± 10 deg/s (Fig. 22c). Figures 22a and 22c clearly show that the aircraft is not able to track the sideslip-angle and roll-rate commands. During the flat turn, the bank angle was kept, with one minor exception, within the adequate range for both legs of the maneuver (Fig. 22d). Note the aircraft is executing turns between 325 and 350 s and between 370 and 395 s to stay within the test range.

A different behavior is observed for the flat turn at -8 deg of angle of sideslip; see

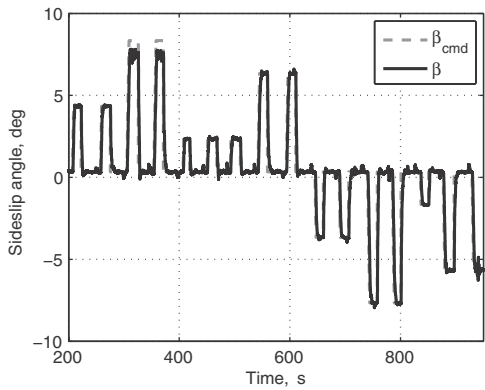


FIG. 21 Angle-of-sideslip vane calibration flight.

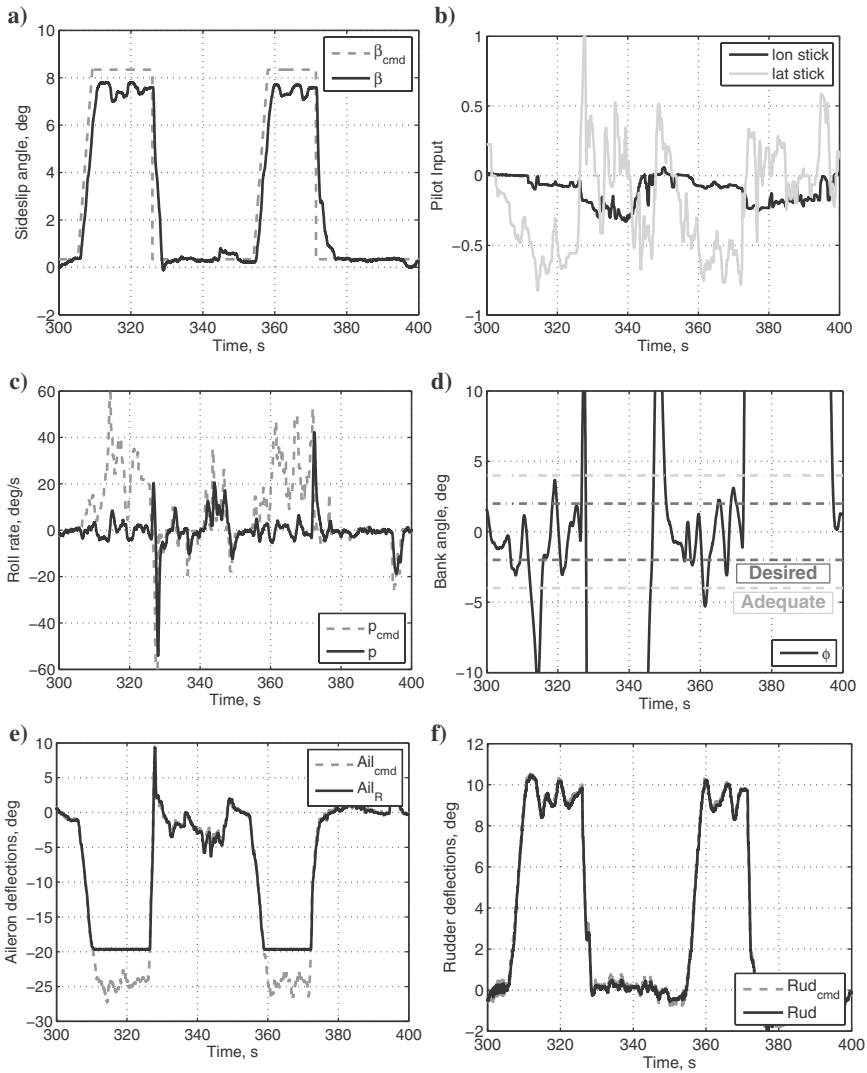


FIG. 22 Angle-of-sideslip vane calibration and flat turn at +8-deg commanded sideslip angle: a) angle-of-sideslip response, b) pilot stick input, c) roll-rate response, d) bank-angle response, e) aileron response, and f) rudder response.

Fig. 23. In this case, the lateral control authority of the GTM seems to be enough to maintain the flat turn. In fact, the \mathcal{L}_1 flight control law was able to provide precise tracking of the commanded steady-state angle of sideslip with roll rates

below ± 5 deg/s, as shown in Fig. 23a and 23c. The bank angle remained within adequate range during the sideslip-angle buildup and within desired range during sideslip-angle hold as seen from Fig. 23d. The pilot stick inputs to maintain flat turn are plotted in Fig. 23b, while the actuator responses are shown in Figs. 23e and 23f.

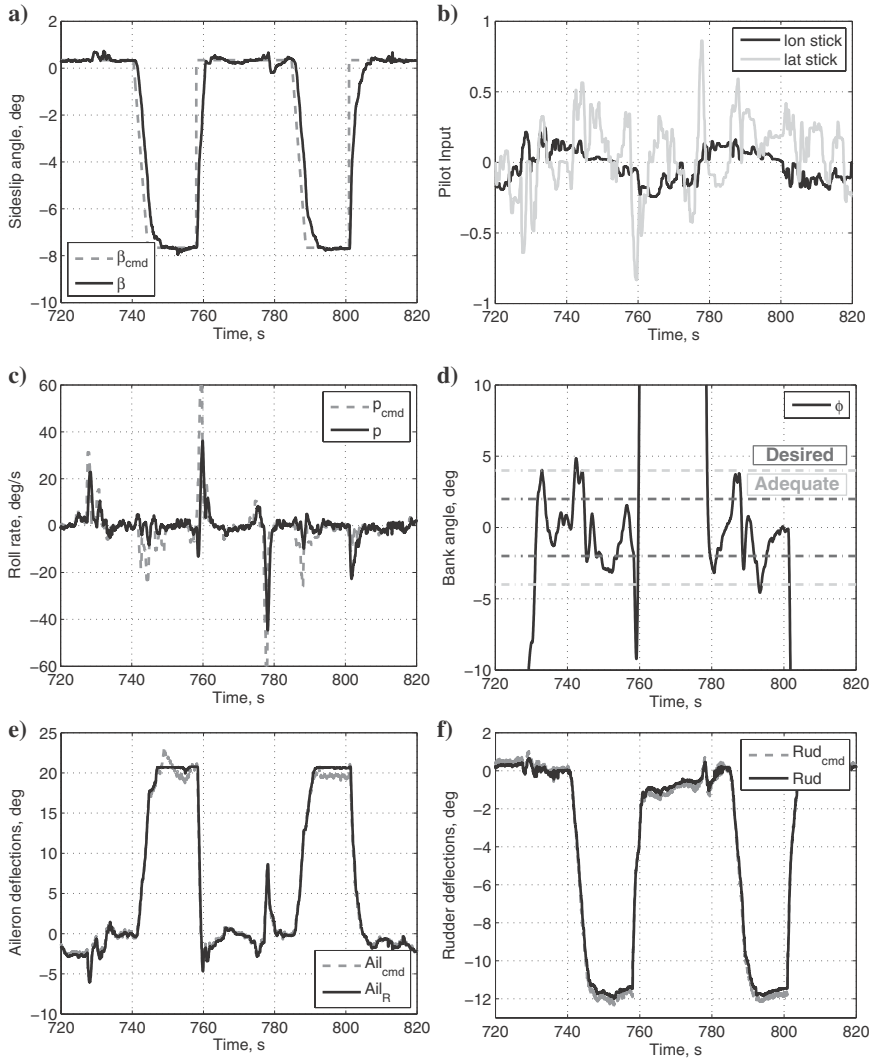


FIG. 23 Angle-of-sideslip vane calibration and flat turn at -8 -deg commanded sideslip angle: a) sideslip-angle response, b) pilot stick input, c) roll-rate response, d) bank-angle response, e) aileron response, and f) rudder response.

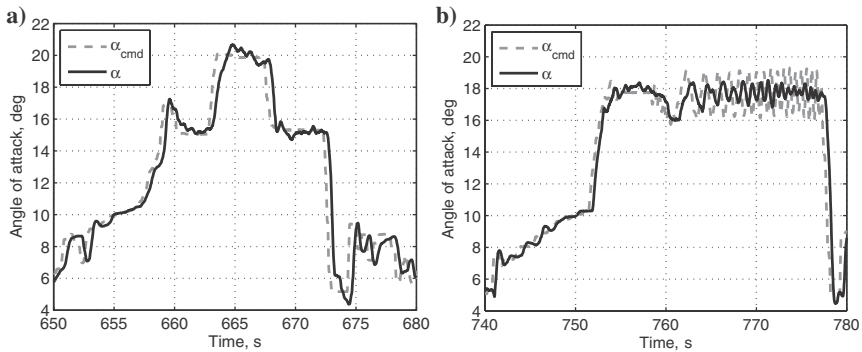


FIG. 24 \mathcal{L}_1 flight control law in support of unsteady aerodynamic modeling work in poststall regimes: a) multistep command and b) Schroeder sweep.

2. UNSTEADY AERODYNAMIC MODELING WORK

During the September 2010 deployment, the \mathcal{L}_1 flight control law was also used to support unsteady aerodynamic modeling work in poststall flight. For this modeling work, the main objective of the control law is to stabilize the aircraft at an angle of attack of 18 deg and, once the desired angle of attack is reached, track a predetermined input wavetrain with the pilot's hands off the stick. Flight-test results for two of these modeling tasks are shown in Fig. 24. In particular, Fig. 24a illustrates a precision tracking task in which the GTM is required to track a multistep wavetrain in the poststall regime, while Fig. 24b presents the response of the system to a Schroeder sweep, also in the poststall regime. As shown in Fig. 24, the \mathcal{L}_1 flight control law provides high angle-of-attack tracking capabilities and ensures a safe recovery once the poststall maneuver is completed. Further description of the performance of the \mathcal{L}_1 flight control law on these and other unsteady aerodynamic modeling tasks can be found in [16].

3. EXPLORATION OF DEPARTURE-PRONE EDGES OF THE FLIGHT ENVELOPE

Finally, during the May 2011 deployment, the \mathcal{L}_1 flight control law continued to support nonlinear unsteady aerodynamic modeling beyond the linear flight regime and also enabled exploration of departure-prone edges of the flight envelope. The \mathcal{L}_1 control law was used as a means to maintain departure resistance and to reduce pilot workload during test-point acquisition. The \mathcal{L}_1 control law allowed the research pilot to operate the aircraft in precarious flight conditions near stall and departure for longer periods of time than direct stick-to-surface aircraft control, which provided enough time for the optimized multi-input wavetrains to excite the aircraft dynamics in all six degrees of freedom and collect the data needed for real-time dynamic modeling. Note that direct stick-to-surface

control is the only control mode, in addition to the \mathcal{L}_1 control law, that is cleared for stall and poststall flight. The \mathcal{L}_1 controller operated without introducing high correlations in the states and controls during the maneuvers. The maneuvers were implemented by adding wavetrain-perturbation inputs directly to the actuator commands from the control law.

The main objective of these research tasks was to collect data and demonstrate real-time stability- and control-parameter identification in numerous off-nominal flight conditions, including stall, poststall, departure, and recovery. The range of angle of attack and angle of sideslip covered for these tasks during flights #54, #55, and #58 is shown in Fig. 25. Of particular interest is the range from 22 to 28 deg of angle of attack, where loss of control is predicted from the nonlinear wind-tunnel based aerodynamic models. In fact, this region is characterized by limited dynamic aircraft controllability, and an abrupt nose-slice departure from controlled flight is expected to occur at around 24-deg angle of attack. During the flights, the \mathcal{L}_1 flight control law was used to slowly acquire and capture a poststall angle of attack in this region of the flight envelope. The \mathcal{L}_1 control law enabled the aircraft to linger in this departure-prone region and made the recovery postdeparture more predictable with lower workload. This maneuver was repeated several times, both with and without the injection of orthogonal multi-sine control surface perturbations. In particular, the flights confirmed the expected nose-slice departure at an angle of attack of 24 deg.

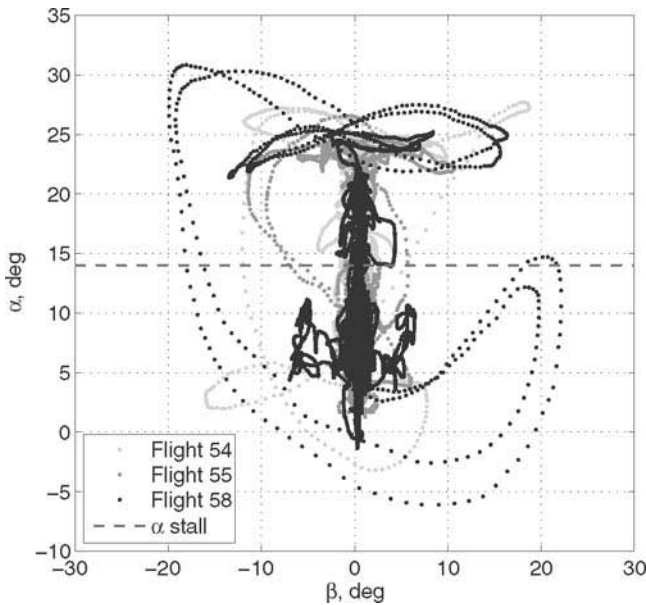


FIG. 25 Angle of attack vs angle of sideslip coverage during large envelope modeling.

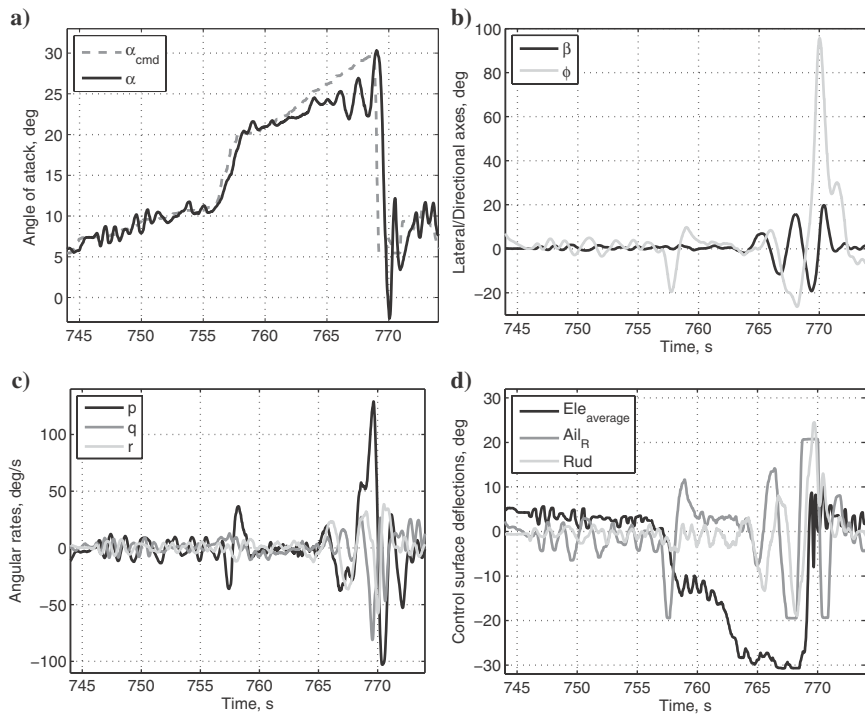


FIG. 26 Aircraft response to angle-of-attack buildup to stall, poststall, departure, and recovery while excited by multi-axis control surface wavetrains: a) true angle of attack, b) sideslip and bank angles, c) angular rates, and d) control surface response.

Figure 26 shows one of the maneuvers used for real-time dynamic modeling in the low angle-of-attack regime, the moderate angle-of-attack regime, stall and poststall flight, departure, and recovery. Figure 26a illustrates the angle-of-attack buildup from wings-level flight at 5-deg angle of attack to lingering at prestall at an angle of attack of about 11 deg, followed to progression through stall and ramping up command to 30 deg. Beginning of departure at around 765 s is evident from both the lateral/directional response and large amplitude angle-of-attack oscillations; see Figs. 26a and 26b. Based on persistent elevator saturation (Fig. 26d), the maximum angle of attack at which the aircraft can be trimmed is approximately 23 deg. Moreover, limitations in control power and vehicle partial controllability become evident from persistent saturation in ailerons starting at 767 s and near saturation of rudder during the same time frame. Full-blown departure with large change in angle of attack, very large values of bank angle, and large roll rates occurs at around 770 s. After the departure, the recovery was immediately initiated and successfully completed as can be seen from all rates

and attitudes returning to nominal conditions. Note that during this entire maneuver there were control-surface perturbations superimposed on the control-surface commands generated by the \mathcal{L}_1 control law. These control-surface excitations can be seen in Fig. 26d as small oscillatory signals on elevator, aileron, and rudder deflections.

D. AIRSTAR FLIGHT-TEST SUMMARY

The series of flight tests on the GTM remotely piloted aircraft conducted with the \mathcal{L}_1 flight control law provided a challenging, real-world environment that stressed the control law to its limits, exposed its behavior in various demanding scenarios, and verified the theoretical predictions of its behavior. The results show that the \mathcal{L}_1 control law provides predictable behavior both in the neighborhood of the design point as well as at other operating points of the flight envelope, such as stall and poststall regimes. Significantly, the control law demonstrates precision tracking capabilities at the edge of aircraft controllability, while the input saturation protection ensures aircraft stability even with persistently saturated control surfaces (occurred during poststall flight and air-data vane calibration).

Fast adaptation is critical for ensuring safe operation and providing predictable aircraft behavior for the pilot, even as significant changes in aircraft dynamics happen unexpectedly. Moreover, the \mathcal{L}_1 controller enables modeling of unsteady nonlinear aerodynamic phenomena at challenging flight conditions, such as stall and poststall, and also enables real-time dynamic modeling of the departure-prone edges of the flight envelope. In this sense, the \mathcal{L}_1 flight control law provides a reliable tool for enhancing the experimental capabilities of the GTM aircraft, by facilitating precision tracking with reduced pilot workload.

IV. CONCLUDING REMARKS AND FUTURE RESEARCH DIRECTIONS

The flight-test results just presented illustrate the performance and robustness characteristics of \mathcal{L}_1 adaptive control. The main features of \mathcal{L}_1 control architectures, proven in theory and consistently verified in these flight tests, can be summarized as follows:

1. Guaranteed robustness in the presence of fast adaptation;
2. Decoupling (separation) between adaptation and robustness;
3. Adaptation rate subject only to hardware constraints;
4. Guaranteed transient response for input and output signals, *without* resorting to persistency-of-excitation-type assumptions, high-gain feedback, and gain-scheduling of the controller parameters (adaptation rate and low-pass filter);
5. Guaranteed (bounded-away-from-zero) time-delay margin for arbitrarily fast adaptation;

6. Uniform *scaled transient response* dependent on changes in initial conditions, unknown parameters, and reference inputs;
7. Computationally predictable numerical characteristics; and
8. Systematic design guidelines.

Some of these features are discussed in more detail in [7], where the interested reader can also find the theoretical developments for the two \mathcal{L}_1 control architectures used in the flight tests with NPS and NASA, as well as other important extensions of the theory. With the features just outlined, \mathcal{L}_1 adaptive control provides a suitable framework for the design of adaptive controllers with performance and robustness guarantees.

While the flight tests presented here demonstrate the advantages of \mathcal{L}_1 adaptive control as a robust adaptive flight control technology, new efforts are needed before it can be transitioned to general aviation. Manned flight brings with it additional challenges. In this regard, experiments are currently being conducted at the Delft University of Technology (The Netherlands) to assess the ability of an \mathcal{L}_1 adaptive flight control system implemented onboard a small business jet to provide enhanced handling qualities and maneuverability margins in the presence of aircraft failures and in different atmospheric conditions. Preliminary results of these experiments can be found in [36]. The study is of particular interest, as it is the first time a rigorously designed \mathcal{L}_1 flight control system is tested on a six-degrees-of-freedom motion-based simulator.

Moreover, in order to ensure postfailure safe flight, additional efforts are also needed to develop real-time schemes for *flight-envelope determination and protection*. In fact, the development of such schemes as well as their integration into a fault-tolerant \mathcal{L}_1 adaptive flight control system are topics of special interest.

ACKNOWLEDGMENTS

The work on Rapid Flight Test Prototyping System at the Naval Postgraduate School was supported by USSOCOM under Tactical Network Topology grant, Office of Naval Research, Air Force Office of Scientific Research, Army Research Office, NASA, and European Commission. The work on AirSTAR was sponsored by NASA. The authors would also like to acknowledge the staff of the AirSTAR Flight Test Facility for their support with control law implementation, as well as for their insights into flight dynamics and piloted evaluations.

REFERENCES

- [1] Anderson, B. D. O., "Failures of Adaptive Control Theory and Their Resolution," *Communications in Information and Systems*, Vol. 5, No. 1, 2005, pp. 1–20.
- [2] Anderson, B. D. O., and Dehghani, A., "Challenges of Adaptive Control—Past, Permanent and Future," *Annual Reviews in Control*, Vol. 32, No. 2, 2008, pp. 123–135.

- [3] Jacklin, S. A., Lowry, M. R., Schumann, J. M., Gupta, P. P., Bosworth, J. T., Zavala, E., Kelly, J. W., Hayhurst, K. J., Belcastro, C. M., and Belcastro, C. M., "Verification, Validation, and Certification Challenges for Adaptive Flight-Critical Control System Software," AIAA Paper 2004–5258, Aug. 2004.
- [4] Wise, K. A., Lavretsky, E., and Hovakimyan, N., "Adaptive Control in Flight: Theory, Application, and Open Problems," *American Control Conference*, IEEE, Piscataway, NJ, 2006, pp. 5966–5971.
- [5] Jacklin, S. A., "Closing Certification Gaps in Adaptive Flight Control Software," AIAA Paper 2008–6988, Aug. 2008.
- [6] Foster, J. V., Cunningham, K., Fremaux, C. M., Shah, G. H., Stewart, E. C., Rivers, R. A., Wilborn, J. E., and Gato, W., "Dynamics Modeling and Simulation of Large Transport Airplanes in Upset Conditions," AIAA Paper 2005–5933, Aug. 2005.
- [7] Hovakimyan, N., and Cao, C., *\mathcal{L}_1 Adaptive Control Theory*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2010.
- [8] Hovakimyan, N., Cao, C., Kharisov, E., Xargay, E., and Gregory, I. M., " \mathcal{L}_1 Adaptive Control for Safety-Critical Systems," *IEEE Control Systems Magazine*, Vol. 31, No. 5, Oct. 2011, pp. 54–104.
- [9] Patel, V. V., Cao, C., Hovakimyan, N., Wise, K. A., and Lavretsky, E., " \mathcal{L}_1 Adaptive Controller for Tailless Unstable Aircraft in the Presence of Unknown Actuator Failures," AIAA Paper 2006–6314, Aug. 2007.
- [10] Kharisov, E., Gregory, I. M., Cao, C., and Hovakimyan, N., " \mathcal{L}_1 Adaptive Control for Flexible Space Launch Vehicle and Proposed Plan for Flight Validation," AIAA Paper 2008–7128, Aug. 2008.
- [11] Leman, T., Xargay, E., Dullerud, G., and Hovakimyan, N., " \mathcal{L}_1 Adaptive Control Augmentation System for the X-48B Aircraft," AIAA Paper 2009–5619, Aug. 2009.
- [12] Michini, B., and How, J., " \mathcal{L}_1 Adaptive Control for Indoor Autonomous Vehicles: Design Process and Flight Testing," AIAA Paper 2009–5754, Aug. 2009.
- [13] Lei, Y., Cao, C., Cliff, E. M., Hovakimyan, N., Kurdila, A. J., and Wise, K. A., " \mathcal{L}_1 Adaptive Controller for Air-Breathing Hypersonic Vehicle with Flexible Body Dynamics," *American Control Conference*, IEEE, Piscataway, NJ, 2009, pp. 3166–3171.
- [14] Gregory, I. M., Cao, C., Xargay, E., Hovakimyan, N., and Zou, X., " \mathcal{L}_1 Adaptive Control Design for NASA AirSTAR Flight Test Vehicle," AIAA Paper 2009–5738, Aug. 2009.
- [15] Gregory, I. M., Xargay, E., Cao, C., and Hovakimyan, N., "Flight Test of \mathcal{L}_1 Adaptive Control on the NASA AirSTAR Flight Test Vehicle," AIAA Paper 2010–8015, Aug. 2010.
- [16] Gregory, I. M., Xargay, E., Cao, C., and Hovakimyan, N., "Flight Test of \mathcal{L}_1 Adaptive Control Law: Offset Landings and Large Flight Envelope Modeling Work," AIAA Paper 2011–6608, Aug. 2011.
- [17] Dobrokhodov, V., Yakimenko, O., Jones, K. D., Kaminer, I., Bourakov, E., Kitsios, I., and Lizarraga, M., "New Generation of Rapid Flight Test Prototyping System for Small Unmanned Air Vehicles," AIAA Paper 2007–6567, Aug. 2007.
- [18] Dobrokhodov, V., Kitsios, I., Kaminer, I., Jones, K. D., Xargay, E., Hovakimyan, N., Cao, C., Lizarraga, M. I., and Gregory, I. M., "Flight Validation of Metrics Driven \mathcal{L}_1 Adaptive Control," AIAA Paper 2008–6987, Aug. 2008.

- [19] Kitsios, I., Dobrokhodov, V., Kaminer, I., Jones, K. D., Xargay, E., Hovakimyan, N., Cao, C., Lizárraga, M. I., Gregory, I. M., Nguyen, N. T., and Krishnakumar, K. S., "Experimental Validation of a Metrics Driven \mathcal{L}_1 Adaptive Control in the Presence of Generalized Unmodeled Dynamics," AIAA Paper 2009-6188, Aug. 2009.
- [20] Li, Z., Dobrokhodov, V., Xargay, E., Hovakimyan, N., and Kaminer, I., "Development and Implementation of \mathcal{L}_1 Gimbal Tracking Loop Onboard of Small UAV," AIAA Paper 2009-5681, Aug. 2009.
- [21] Kaminer, I., Pascoal, A., Xargay, E., Hovakimyan, N., Cao, C., and Dobrokhodov, V., "Path Following for Unmanned Aerial Vehicles Using \mathcal{L}_1 Adaptive Augmentation of Commercial Autopilots," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 550-564.
- [22] Dobrokhodov, V., Kaminer, I., Kitsios, I., Xargay, E., Hovakimyan, N., Cao, C., Gregory, I. M., and Valavani, L., "Experimental Validation of \mathcal{L}_1 Adaptive Control: The Rohrs Counterexample in Flight," *Journal of Guidance, Control and Dynamics*, Vol. 34, No. 5, 2011, pp. 1311-1328.
- [23] Pomet, J. B., and Praly, L., "Adaptive Nonlinear Regulation: Estimation from the Lyapunov Equation," *IEEE Transactions on Automatic Control*, Vol. 37, No. 6, June 1992, pp. 729-740.
- [24] Cichella, V., Xargay, E., Dobrokhodov, V., Kaminer, I., Pascoal, A. M., and Hovakimyan, N., "Geometric 3D Path-Following Control for a Fixed-Wing UAV on $SO(3)$," AIAA Paper 2011-6415, Aug. 2011.
- [25] Cunningham, K., Foster, J. V., Morelli, E. A., and Murch, A. M., "Practical Application of a Subscale Transport Aircraft for Flight Research in Control Upset and Failure Conditions," AIAA Paper 2008-6200, Aug. 2008.
- [26] Murch, A. M., "A Flight Control System Architecture for the NASA AirSTAR Flight Test Infrastructure," AIAA Paper 2008-6990, Aug. 2008.
- [27] Cunningham, K., "AirSTAR Flight Test Plan: 5.5% Dynamically Scaled GTM Tail Number T2. Deployment: 2010.02," NASA Technical Rept. GTMP-6326 2010.02, V 2.00, May 2010.
- [28] Cunningham, K., "AirSTAR Flight Test Cards, v3.12," Technical Rept. GTMP-6325 2010.03, V 3.12, Aug. 2010.
- [29] Cooper, G. E., and Harper, R. P., Jr., "The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities," NASA, Technical Note D-5153, April 1969.
- [30] "Flying Qualities of Piloted Aircraft," Interface Standard, U.S. Dep. of Defense Military Specification, MIL-STD-1797B, Washington, D.C., Feb. 2006.
- [31] Xargay, E., Hovakimyan, N., and Cao, C., " \mathcal{L}_1 Adaptive Controller for Multi-Input Multi-Output Systems in the Presence of Nonlinear Unmatched Uncertainties," *American Control Conference*, IEEE, Piscataway, NJ, June-July 2010.
- [32] Krstić, M., Kanellakopoulos, I., and Kokotović, P. V., *Nonlinear and Adaptive Control Design*, Wiley, New York, 1995, pp. 87-121.
- [33] Yao, B., and Tomizuka, M., "Adaptive Robust Control of SISO Nonlinear Systems in a Semi-Strict Feedback Form," *Automatica*, Vol. 33, No. 5, 1997, pp. 893-900.
- [34] Cunningham, K., Cox, D. E., Murri, D. G., and Riddick, S. E., "A Piloted Evaluation of Damage Accommodating Flight Control Using a Remotely Piloted Vehicle," AIAA Paper 2011-6451, Aug. 2011.

- [35] Lawford, J. A., and Nippres, K. R., "AGARD Flight Test Techniques Series. Volume 1 on Calibration of Air-Data Systems and Flow Direction Sensors," (NATO) Advisory Group for Aerospace Research and Development, Technical Note AGARD-AG-300-VOL.I, Neuilly-sur-Seine, France, Sept. 1983.
- [36] Stroosma, O., Damveld, H. J., Mulder, J. A., Choe, R., Xargay, E., and Hovakimyan, N., "A Handling Qualities Assessment of a Business Jet Augmented with an \mathcal{L}_1 Adaptive Controller," AIAA Paper 2011-6610, Aug. 2011.

Integrated Systems Health Management for Intelligent Systems

Fernando Figueroa*

NASA Stennis Space Center, Mississippi 39529

Kevin Melcher†

NASA John H. Glenn Research Center at Lewis Field, Cleveland, Ohio 44135

ACRONYMS

CBM	condition-based maintenance
CSG	chemical steam generator
DIaK	data, information, and knowledge
DIaKA	data, information, and knowledge architecture
FCL	functional capability level
FMEA	failure modes and effects analysis
GN	gaseous nitrogen
GRC	Glenn Research Center
IMBT	ISHM Model Building Toolkit
IPA	isopropyl alcohol
IS	intelligent sensor
ISHM	Integrated Systems Health Management
ISHM-DM	ISHM domain model
KSC	Kennedy Space Center
LC-20	Launch Complex 20
LOX	liquid oxygen
NCAP	Network Capable Application Processor
NIST	National Institute of Standards and Testing
OSA-CBM	Open Systems Architecture for Condition-Based Maintenance
RCA	root cause analysis
RETS	rocket engine test stand
SS	smart sensor
SSC	Stennis Space Center

*Innovative Partnerships Division. Associate Fellow AIAA.

†Senior Research Engineer, RHC/Controls & Dynamics Branch, 21000 Brookpark Road, Mail Stop 77-1. Senior Member AIAA.

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

SS&As	smart sensors and actuators
STE	special test equipment
S4	Systematic Sensor Selection Strategy
S&As	sensors and actuators
TCP/IP	transmission control protocol/Internet protocol
TEDS	transducer electronic data sheet
TIM	transducer interface module
WISE	Virtual Intelligent Sensor Environment

I. INTRODUCTION

In this chapter, Integrated Systems Health Management (ISHM) is presented as an enabling discipline/technology area for intelligent systems, as well as a capability that embodies “intelligence” in itself. To that end, a variety of intelligent systems-relevant ISHM topics are addressed, including examples. The information presented provides the reader with an understanding of the state-of-the-art current research and challenges that are relevant to ISHM as a core capability of an intelligent system. (To distinguish ISHM for intelligent systems from more typical approaches to implement ISHM capability, the authors will use the term *Intelligent ISHM* or *iISHM*.) The following definitions are provided to ensure accurate interpretation of the manuscript:

Data: This information comprises facts, observations, or perceptions (which may or may not be correct). Alone, data represent raw numbers or assertions and may therefore be devoid of context, meaning, or intent [1].

Engineering data: These are data that have been given an offset or amplification or data with engineering units.

Information: This includes data that possess context, relevance, and purpose. Information typically involves the manipulation of raw data to obtain a more meaningful indication of trends or patterns in the data [1]. Information pertains to known facts that result from processed data, and inferred facts from data by using knowledge and other facts.

Knowledge: Knowledge in an area is defined as justified beliefs about relationships among concepts relevant to that particular area. This is intrinsically different from information [1]. Knowledge consists of facts and inference rules used for reasoning. Knowledge can be procedural, which refers to knowing how to do something, or declarative, which refers to knowing that something is true or false. . . . We say we can access information, but only an agent possesses knowledge. Traditionally, knowledge has been associated with the concept of belief, which refers to statements that are inside the mind of an agent or can be inferred by the agent. These statements do not have to be true and can be believed to varying degrees [2].

Intelligent sensor (IS): This sensor is a sensor that is network capable (can communicate over a network) and provides data, a measure of the quality of the data, and a measure of the health condition of the sensor itself. The IS

integrates data, information, and knowledge (DIaK) local to it and from other elements in the network in order to achieve a maximum “degree of intelligence” (DoI). An IS with the highest DoI provides a complete and accurate measure of the quality of the data and a complete and accurate assessment of its health. An IS may achieve its functionality by a combination of hardware and software elements, some of which may be located close to the sensor element (or transducer) and some at a distance. The IS may include the ability to self-reconfigure to overcome anomalies in its operation.

ISHM has been defined from many perspectives. Here iISHM it is defined as a capability achieved by integrating data, information, and knowledge (DIaK) that is conceptually and/or physically distributed throughout the system elements (which inherently implies the capability to manage DIaK associated with distributed subsystems). As used here, management of DIaK encompasses contextual and timely storage, distribution, sharing, maintenance, processing, reasoning, and presentation. This paradigm implies that DIaK must be available to any element of a system at the right time and in accordance with a meaningful context. iISHM functional capability level (FCL) is measured by how well a system performs the following functions: 1) detect anomalies, 2) diagnose causes, 3) predict future anomalies/failures, 4) enable efficient integration and execution of all phases of the life cycle of a system from a systems engineering perspective, and 5) provide the user with an integrated awareness about the condition of important elements of the system as a means of guiding user decisions.

The chapter is organized as follows: Sec. II describes core areas of iISHM capability development, including standards and related architectures, the iISHM knowledge model, software tools, intelligent sensors and components, and sensor selection and placement. Section III describes iISHM in the context of systems design, integration, and engineering. Section IV briefly describes controls for iISHM-enabled systems. Section V briefly describes opportunities for advances in validation and verification of iISHM systems. Section VI includes a detailed implementation of iISHM capability, and Sec. VII provides conclusions and recommendations on the way forward.

II. iISHM CAPABILITY DEVELOPMENT

iISHM functions are currently performed manually. For complex systems, it involves many people; it is very costly and difficult to improve with time and use; it involves minimal integration of DIaK across the system; it is not comprehensive (does not include all elements of a system or much DIaK about the system); and it is not continuous (a people-based system is generally not vigilant 24 hours a day, every day). Figure 1 describes the layered approach currently employed to achieve iISHM capability. At the top layer, layer 1, onboard iISHM capability is deployed. At the moment, this amounts to monitoring thresholds on a few sensor measurements in order to avoid catastrophic events. In layer 1, only

People-Based ISHM is Being Done Today

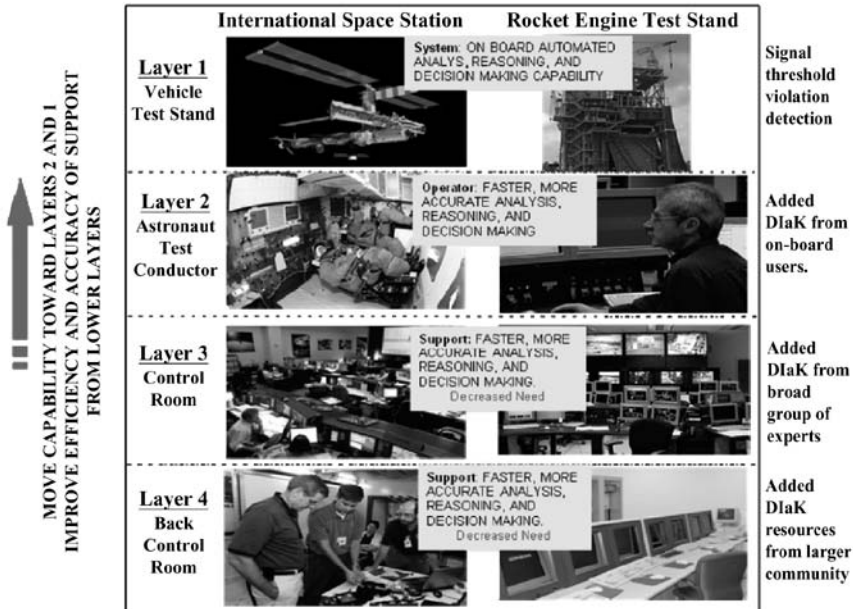


FIG. 1 Description of a layered iISHM capability as is done today.

events are detected, which are used by people in the lower layers to reason and infer what the associated anomaly might be (diagnose its causes, etc.). Layers 2 to 4 involve people in increasing numbers and even entire organizations aligned with individuals in the control/evaluation rooms. iISHM capability onboard the system is expected to enable faster and more accurate analysis, reasoning, and decision making in layers 2–4.

There are a number of implementations of ISHM capability since before 2000. NASA GRC led some advances in health management for propulsion systems [3]. These efforts included a combination of sensor validation methods and expert systems for diagnosis, applied to rocket engine test postdiagnostics. Another example is Boeing's 777 Airplane Health Management (AHM) system [4]. AHM involves a central maintenance computer that collects information from many sub-systems that encompass built-in test elements. AHM's purpose was to decrease unplanned maintenance from the 75% level to a 25% level. This degree of improvement could only be achieved by augmenting the knowledge and information base in each subsystem, through processing and reasoning across subsystems. This strategy was supported by the AHM architecture. Boeing and Pratt & Whitney Rocketdyne implemented the Advanced Health Management System (AHMS) in the space

shuttle main engine (SSME) [5]. AHMS was developed to meet more stringent engine reliability requirements. The long-term goals were ambitious, encompassing an integrated approach to detect anomalies, diagnose causes, and predict future anomalies; however, only a first phase was implemented, encompassing monitoring of vibration sensors mounted on the high-pressure fuel turbopump and high-pressure oxidizer turbopump. AHMS was certified and used in flight, with the authority to shut down the engine.

Another attempt to advance ISHM capability implementation was embodied by NASA's Propulsion IVHM Technology Experiment (PITEX), where IVHM stands for Integrated Vehicle Health Management. PITEX implemented an architecture that represented system elements with state models and used states of system parameters to reason and make decisions about the health of the elements. PITEX used the software environment Livingstone to model a propulsion system composed of tanks, valves, and other basic elements; it was tested using simulated data [6].

A system that has been in use for a long time is the Health and Usage Monitoring System (HUMS) for helicopters (a web search will uncover a large number of references). The HUMS monitors data from helicopter subsystems and processes it using a set of specialized algorithms. The resulting anomaly indicators and original data are used by experts to infer if critical elements might be trending toward failure. In this system, knowledge and its integrated interpretation is primarily done by people.

Although the references just cited implement ISHM to some level of capability, they do not represent "intelligent" implementations that would encompass embedded DIaK, nor embrace intelligent systems architectures, paradigms, or ontologies. The following sections describe technologies, tools, and infrastructure needed to achieve ISHM capability that is mainly onboard the system, is affordable and evolutionary throughout the life of the system, integrates DIaK across the system, is continuous, and is comprehensive. To make this possible, it is necessary that the ISHM capability incorporate a knowledge-based approach and hence embody "intelligence."

A. STANDARDS FOR iISHM IMPLEMENTATION

The development of an iISHM capability requires the use of models (knowledge) applied to information and data associated with various elements that make up a system. Here, the term "model" is used in the broadest sense as it may include qualitative (e.g., heuristics), analytic, statistical, fuzzy-logic, classic logic, artificial neural network, and other types of models. Use of models is enabled by management of DIaK, encompassing storage, distribution, sharing, maintenance, processing, reasoning, and presentation. To make this possible in a generic manner, beyond specific applications; and to achieve greater affordability, standards must be established so that DIaK can be managed in a plug-and-play and interoperable manner.

Standards for iISHM must be at a high enough layer in the infrastructure so that they are largely independent of the physical (e.g., Ethernet) and transmission (e.g., TCP/IP) layers. Example standards for iISHM include the Institute of Electrical and Electronics Engineers (IEEE) 1451 family of standards for smart sensors and actuators, the Open Systems Architecture for Condition-Based Maintenance (OSA-CBM) standard, and the Open Systems Architecture for Enterprise (OSA-EAI) standard managed by the Machine Information Management Open Standards Alliance (MIMOSA). These standards are sufficiently abstracted so that they can be implemented utilizing any physical or transmission architecture.

1. IEEE 1451 FAMILY OF STANDARDS FOR SMART SENSORS AND ACTUATORS (SS&A)

The IEEE 1451 family of standards was developed by government and private entities under the leadership of the National Institute of Standards and Technology (NIST). Reference [7] provides a summary of the standards and their use. In creating these standards, the objective was to standardize DIaK associated with sensors and actuators (S&As). The standards are described as a family because, as evidenced from the quote in the following paragraph, they address various elements and functions of smart sensors and actuators (SS&As). The notion is that SS&As must incorporate DIaK related to their functionality and provide their DIaK, via a communications network, to other systems or functions that use and manage S&As.

The IEEE (Institute of Electrical and Electronics Engineers) 1451 smart transducer interface standards provide the common interface and enabling technology for the connectivity of transducers to microprocessors, control and field networks, and data acquisition and instrumentation systems. The standardized TEDS specified by IEEE 1451.2 allows the self-description of sensors and the interfaces provide a standardized mechanism to facilitate the plug and play of sensors to networks. The network-independent smart transducer object model defined by IEEE 1451.1 allows sensor manufacturers to support multiple networks and protocols. Thus, transducer-to-network interoperability is on the horizon. The inclusion of P1451.3 and P1451.4 to the family of 1451 standards will meet the needs of the analog transducer users for high-speed applications. In the long run, transducer vendors and users, system integrators and network providers can all benefit from the IEEE 1451 interface standards. [8]

The most common physical architectures for systems are bus-based multidrop configurations. Figure 2 shows configurations and implementation of IEEE 1451 standards, and a short summary is provided next. The standards are still being modified, but the intent here is to provide a sense of how the standards can be used to enable interoperability and plug-and-play capability with networked transducers encompassing embedded information (hence, smart transducers).

IEEE P1451.0 defines a set of common commands, common operations, and transducer electronic data sheet (TEDS) for the family of IEEE 1451 standards.

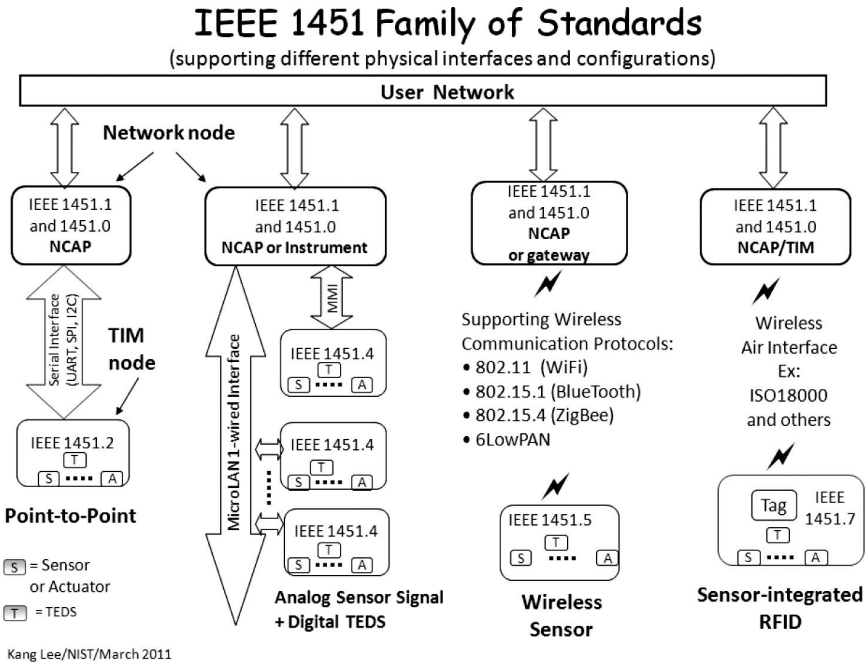


FIG. 2 IEEE 1451 family of standards (courtesy of Dr. Kang Lee, NIST).

The commands allow communication with sensors or actuators in IEEE 1451-based wired and wireless networks. The functionality is independent of the physical communications media and the network node called Network Capable Application Processor (NCAP).

IEEE 1451.1 defines a common object model describing the behavior of smart transducers (sensor and actuators). It defines the communication models used for the standard, which include the client-server and publish-subscribe models. Application software based on IEEE 1451, running in the NCAP, communicate with transducers through any physical layer standards as needed for a particular application. The standard enables communications among NCAPs and to higher level systems, in a network neutral manner.

IEEE 1451.2 defines transducers-to-NCAP interface and TEDS for a point-to-point configuration. Transducers are part of a transducer interface module (TIM). The original standard describes a communication layer based on enhanced SPI (serial peripheral interface) with additional hardware (HW) lines for flow control and timing.

IEEE 1451.3 defines a transducer-to-NCAP interface and TEDS for multidrop transducers using a distributed communications architecture. It allows many

transducers to be arrayed as nodes, on a multidrop transducer network, sharing a common pair of wires.

IEEE 1451.4 defines a mixed-mode interface for analog transducers with analog and digital operating modes. A TEDS was added to a traditional two-wire, constant-current excited sensor containing a FET amplifier. The TEDS model was also refined to include critical information that must fit in a small memory device, needed by very small transducers. Templates are used to describe the data structure of TEDS. The current templates cover accelerometers, strain gauges, current loop sensors, microphones, thermocouples, and others.

IEEE P1451.5 defines a transducer-to-NCAP interface and TEDS for wireless transducers. Wireless communication protocol standards such as 802.11 (WiFi), 802.15.1 (Bluetooth), and 802.15.4 (ZigBee) are being considered as some of the physical interfaces for IEEE P1451.5. The objective is to be able to communicate with a wireless transducer embodying any of these three wireless protocols.

IEEE P1451.6 defines a transducer-to-NCAP interface and TEDS using the high-speed CANopen network interface. Both intrinsically safe and nonintrinsically safe applications are supported. It defines a mapping of the 1451 TEDS to the CANopen dictionary entries as well as communication messages, process data, configuration parameter, and diagnosis information. It adopts the CANopen device profile for measuring devices and closed-loop controllers.

2. OSA-CBM STANDARD

The OSA-CBM standard was developed by government and private entities. This standard addresses management of health information from any element, subsystem, system, or system of systems. The foundation is the definition of layers where health information is organized according to the degree of processing, and hence the amount of DIaK employed, to determine health condition (Fig. 3; see also <http://www.mimosa.org>) [9]. The standard focuses on automated real-time management of health information. In contrast, health management over extended periods of time (non real time) is typically based on large databases and done primarily by people. The non-real-time approach is standardized as the Open Systems Architecture for Enterprise Application Integration (OSA-EAI) standard. Both standards are maintained by the MIMOSA organization (data available online at <http://www.mimosa.org>). MIMOSA “is a non-profit trade association dedicated to developing and encouraging the adoption of open information standards for Operations and Maintenance in manufacturing, fleet, and facility environments. MIMOSA’s open standards enable collaborative asset lifecycle management in both commercial and military applications.”

3. EXAMPLE IMPLEMENTATION OF IEEE 1451 AND OSA-CBM STANDARDS

Figure 4 shows a physical architecture (bus-based, multidrop, Ethernet network) for a pilot iISHM system implemented at NASA Kennedy Space Center, Launch

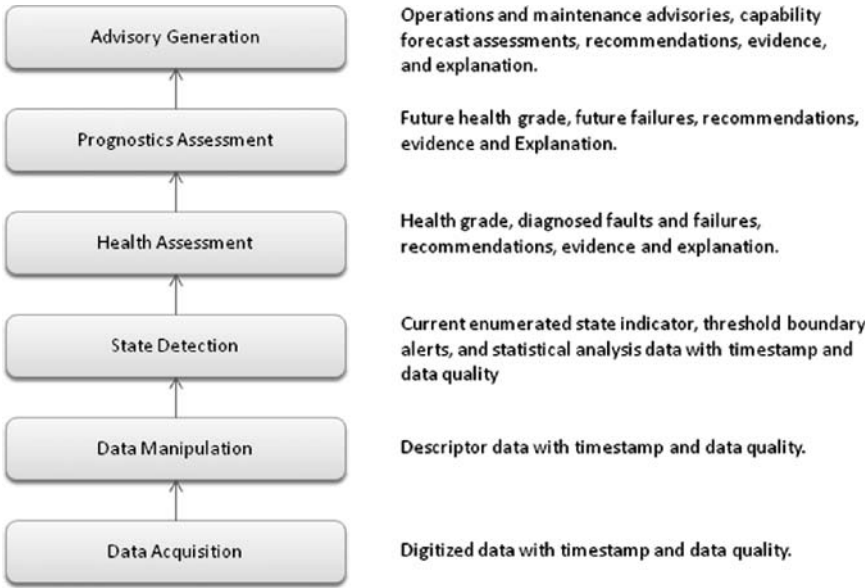


FIG. 3 CBM layered health DIaK architecture.

Complex 20 (LC-20) [9]. The architecture is hierarchical, with buses at various levels, where higher-level information flows up toward the site-wide management computer. This is a typical architecture for systems in most industries, including aerospace. Standards were implemented in the lower part of the physical architecture (bus showing IEEE 1451.1 and OSA-CBM standards on Ethernet-IEEE 802.3), but it was sufficient to demonstrate the impact of standards for iISHM implementation in an operational system, during a test at LC-20.

The experiment demonstrated interoperability of iISHM systems developed by three different providers: NASA Stennis Space Center (NASA SSC), NASA Kennedy Space Center (NASA KSC), and the Pennsylvania State University’s Applied Research Laboratory (PSU-ARL). The interoperability was enabled through the use of the IEEE 1451 and OSA-CBM standards.

B. iISHM DOMAIN MODEL (iISHM-DM)

The concept of an iISHM domain model (iISHM-DM) has been introduced previously [10, 11]. iISHM-DM embodies specific DIaK that are needed to achieve iISHM capability, including system element identification and specifications and interelement relationships used in reasoning approaches. Data are available from sensors and components. Distribution of DIaK associated with the physical elements of a system gives rise to an iISHM architecture that enables distributed management of DIaK to achieve ISHM functionality. The iISHM architecture is a

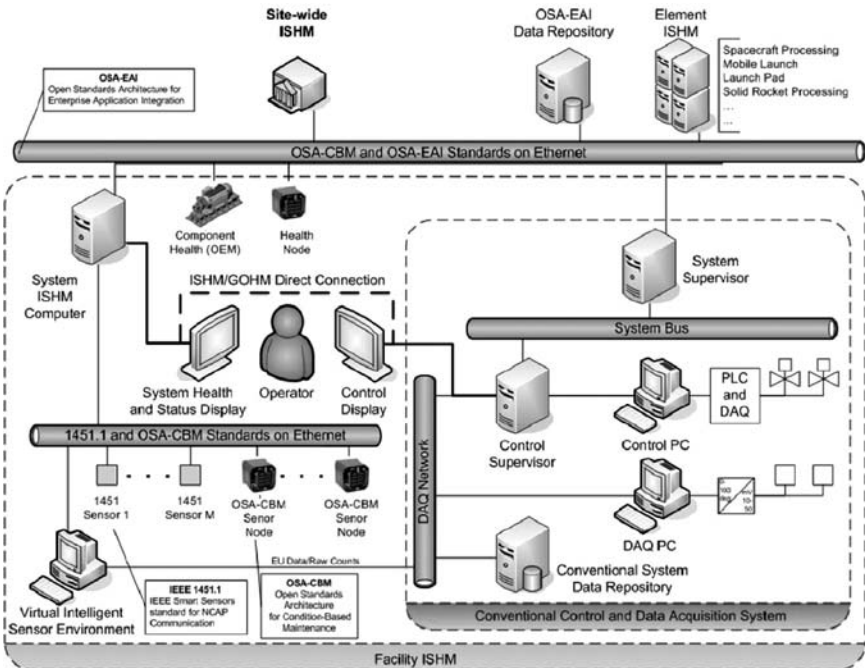


FIG. 4 Architecture for pilot iISHM system implemented at NASA Kennedy Space Center, Launch Complex 20 (LC-20) showing the use of IEEE 1451.1, OSA-CBM, and OSA-EAI standards [9].

DIaK-Architecture (DIaKA), where intelligent processes (e.g., physics-based models) providing various degrees of integration (through interdependencies) are used to achieve the desired iISHM capability and where DIaK are managed in a distributed manner (Fig. 5). This hierarchical architecture enables abstracting models of processes occurring throughout the system (e.g., tank pressurization, subsystem leak, valve leak, sensor flat, etc.) and is conceptually different from a typical architecture depicting the physical composition of a system, where the hierarchy is based on simpler physical elements being assembled into more complex systems and subsystems.

Figure 6 is another depiction of the DIaKA. DIaK are distributed among the elements of a system, including sensors, actuators, and components, as well as subsystems and systems. The icons represent active repositories of data and information pertinent to their function, operation, and health. The icons also represent process models (knowledge) that enable iISHM functionality. Figure 6 shows a representation of the DIaKA as related to an iISHM-DM of a simple rocket test stand system.

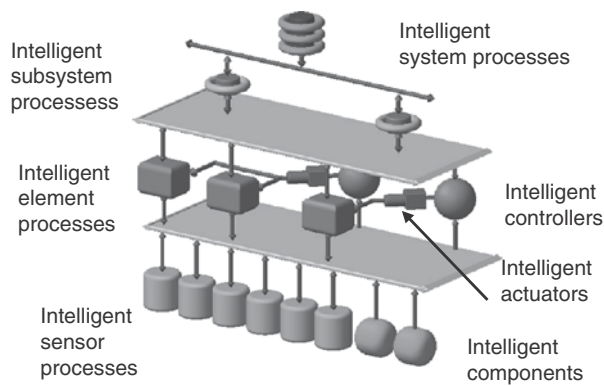


FIG. 5 Abstract representation showing that data, information, and knowledge architecture is both distributed and hierarchical in an iISHM system.

The DIaKA supports the following paradigm. Sensor icons are repositories for sensor processes that operate on measurements within a local context, independent of other elements of the system. Sensor processes are, for example, algorithms to determine level of noise, changes on level of noise, flat signals, time response characteristics, etc. In addition, sensor processes include health

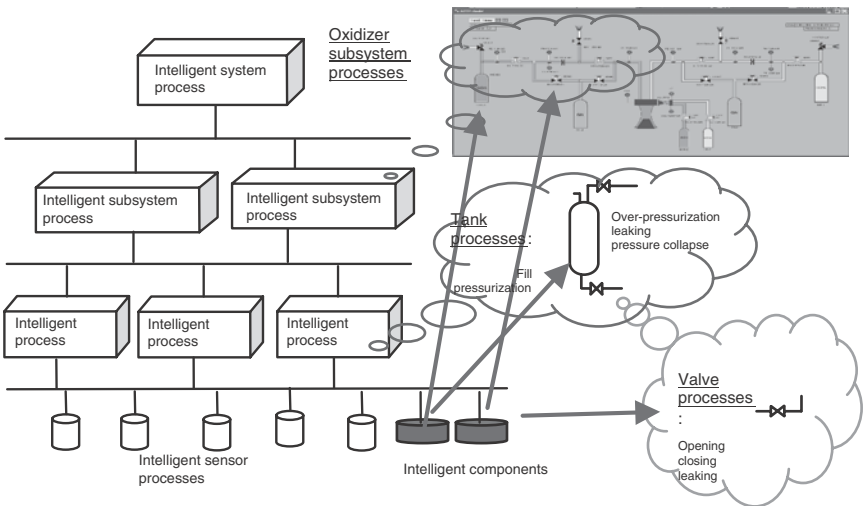


FIG. 6 DIaKA showing the correspondence of practical iISHM-DM elements adapted for a small rocket test stand. Process models are executed often in parallel for consistency checking that leads to anomaly detection and reasoning about health of processes, sensors, and components.

assessment processes focused on determining 1) sensor health and 2) the quality of the measurement. Health assessment sensor processes also receive information from other processes higher in the hierarchy to improve their health assessment. For example, a process model of flow from a tank through a valve, to atmosphere can allow consistency checks among pressure and temperature sensors along the path of the flow. If one sensor is inconsistent with the model, this information is fed back to the sensor to improve its own health assessment and anomaly determination. The same applies to component processes. The final objective is to determine the health of sensors and components and do it with maximum utilization of DIaK embodied in the various layers of processes in the hierarchy [10]. Health is assessed through a cycle that begins at the sensor and component levels, using localized DIaK. Then, data and health information are communicated to the next higher level for consistency checking, using models that describe multiple processes occurring in one or more subsystems that encompass elements in the lower level. Results of consistency checking at this level (health results) are then pushed down to improve assessments at lower levels, but also elevated to the next higher level for consistency checking using process models at that level (richer in information). This cycle completes when data and health information traverse up and down back to the sensor or component, resulting in a health assessment that is the product of consistency checking using all available and applicable DIaK in the iISHM domain model. This paradigm to assess health of sensors and components has only partially been implemented as an automated capability because an enabling software environment for a complete implementation is not yet available.

C. SOFTWARE CAPABILITIES TO DEVELOP iISHM DOMAIN MODELS

Core capabilities of ISHM include 1) detecting anomalies, 2) diagnosing causes, 3) predicting future anomalies/failures, 4) enabling efficient integration and execution of all phases of the life cycle of a system from a systems engineering perspective, and 5) providing the user with an integrated awareness about the condition of important elements of the system as a means of guiding user decisions. These capabilities are to provide continuous and comprehensive awareness about the health of every element of a system. DIaK must be employed to do the reasoning leading to achieving the core capabilities. Furthermore, multiple simultaneous process models and approaches should be employed to achieve maximum functional capability level (FCL). By using all applicable process models at any given time, one makes more effective use of all DIaK embodied in the iISHM-DM. A software system for iISHM capability should support all core capabilities by enabling systematic integration of DIaK throughout the iISHM-DM. The following requirements should be met by the software system:

Object representation: An object-oriented representation of system physical elements and associated process models is the best way to implement DIaK in a

systematic and organized manner. Object-oriented programming methods, one approach used to implement object-oriented representations, provide for clear modularly structured object representations that are easy to maintain and modify as new objects are created and also provide a more intuitive understanding of the code and its outcomes.

Distribution of iISHM-DMs within and across networks: Given the distributed nature of the DIaKA, iISHM-DMs are likely to be distributed among various processors connected to a network. Also, iISHM-DMs might be created by different people in various geographic locations. This necessitates the use of parallel processing, be it using a single computer or using systems across networks. As the complexity of systems increases, and/or a large number of process models are used in achieving effective ISHM capability, it might not be reasonable, or manageable, to do this with a centralized architecture.

Distribution across processing units: Because multiple process models are expected to be running simultaneously, the software environments should support parallel processing.

Inference engine: Many tasks require an inference engine. Reasoning and decision making leading to anomaly detection, diagnostics, effects, and prognostics require contextual integrity and cause-effect analysis using heterogeneous data and information. The inference engine must also allow accurate representation and automatic execution of failure modes and effects analysis (FMEA).

Integrated management of distributed DIaK: DIaK must be managed in a way to allow embodiment of systems thinking across elements and subsystems. Often this is enabled by definitions of relationships among elements of systems that can be physically visible (i.e., attached to, belong to a system) or more abstracted relationships, as it relates to involvement in process models (e.g., pressure sensors associated to a particular subsystem, subsystem definitions that change with configuration, etc.).

Definition of dynamic relationships among objects for use in reasoning: Often, the framework for reasoning and application of process models changes dynamically with configuration changes, stages of operation, etc. This also means that relationships among objects and processes change dynamically and must be represented in the ISHM-DMs. For example, reasoning to detect leaks in a sealed subsystem requires that membership of elements belonging to sealed subsystems must change with valve state changes.

Iconic representation of systems objects with visible and virtual links (relationships) used to provide intuitive representation of reasoning and context: The mix of object representation and iconic representation of DIaK provides the ability to intuitively visualize interrelationships and dig deep into details of the iISHM system. As complexity increases, graphical programming and visualization become essential.

A software environment developed by NASA Stennis Space Center and General Atomics [10–14] meets all of the preceding requirements. The software was developed using G2 (data available online at <http://www.gensym.com>), which is a commercial programming environment for implementation of intelligent applications. Another environment that enables integrated management of DIaK, which could be applied to creation of iISHM domain models (defined in the next section), is Brahms [15]. Brahms is a multi-agent modeling environment for simulating processes and practices. Other software environments for creation of knowledge models include TEAMS (data available online at <http://www.teamqsi.com/index.html>) and MADe (data available online at <http://www.phmtechnology.com/>) for automating (FMEA) and Livingstone for state-machine models [16].

D. INTELLIGENT SENSORS AND COMPONENTS

The lower elements in the DIaKA (Fig. 6) represent processes associated with sensors and components, where “components” is intended to encompass any element that is not a sensor, for example, tanks, pumps, etc. These elements directly represent physical entities in the system, and, in the future, they are expected to incorporate their own embedded processing and networking capabilities. This is already true for sensors, as many “intelligent sensor” concepts are now available commercially, and more are in development (data available online at <http://www.eesensors.com/index.html> and <http://www.mobitrum.com>).

There are many definitions for IS. The following definition is based on the foundation provided by the IEEE 1451 family of standards for smart sensors and actuators. It is reasonable to assume that the standard defines a “smart sensor (SS),” as described earlier in Sec. II.A. “Intelligent sensor” is therefore a “smart sensor” with the ability to provide the following functionality: 1) measurement, 2) assessment of the quality of the measurement, and 3) determination of the “health” of the sensor. The better the sensor provides functionalities 2 and 3, the more intelligent it is.

Implementation of IS can be done in many ways. Commercial SS incorporating TEDS in-a-chip have been available for some time (a web search will reveal many offerings). Some IS or SS modules have been developed in industry. These are small format units that incorporate signal conditioning, data acquisition, processing capability, and protocols for communicating as network elements (data available online at <http://www.eesensors.com/index.html> and <http://www.mobitrum.com>).

In other cases, IS capability is enabled by a combination of hardware and software that turns conventional sensors into smart sensors, as is the case with products from National Instruments that support TEDS [21] (data available online at <http://www.ni.com>). Intelligent sensor functionality has also been implemented purely in software, again, to turn conventional sensors into intelligent ones. Figure 7 shows the configuration of a pilot ISHM implementation for a rocket

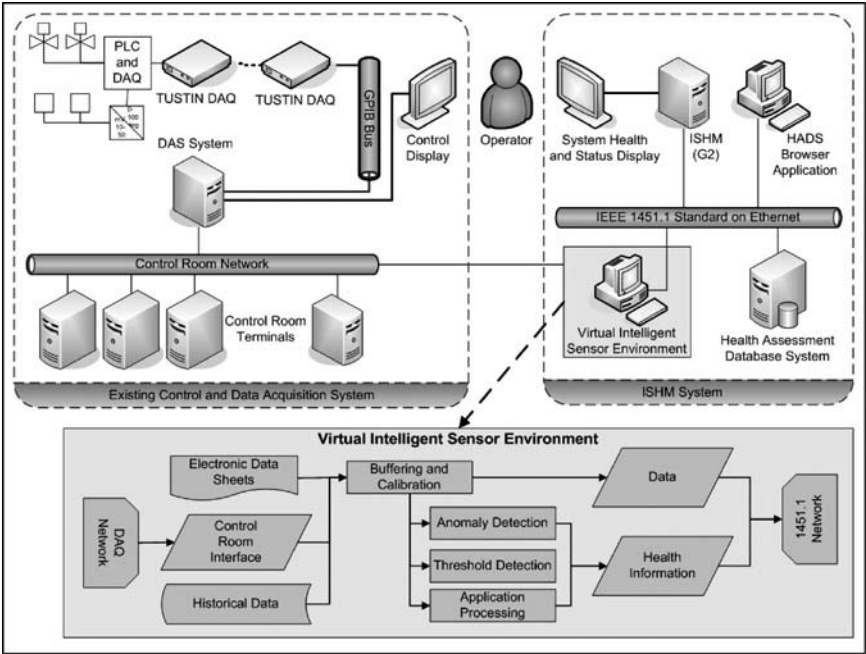


FIG. 7 Architecture showing implementation of intelligent sensors through the Virtual Intelligent Sensor Environment [12].

engine test stand. Here the Virtual Intelligent Sensor Environment (VISE) turns all classic test-stand sensors into intelligent sensors [10]. The VISE publishes IS data and information to a bus for consumption by the ISHM system and other users such as repositories and visualization systems. Some of the processes to be embedded in IS include the following: noise level assessment and history, spike detection and history, flat signal detection and history, response time characterization, intermittency characterization and history, physical detachment characterization and history, regime characterization and history, and curve fit on identified regimes.

E. OPTIMIZING SENSOR SELECTION AND PLACEMENT FOR iISHM

When developing an ISHM capability from the ground up, one must optimize sensor suites to achieve maximum functional capability (e.g., anomaly detection, diagnosis, effects, prognostics). The Systematic Sensor Selection Strategy (S4) [17–21] is one approach that has been developed to provide this important iISHM capability. S4 is a model-based procedure for systematically and

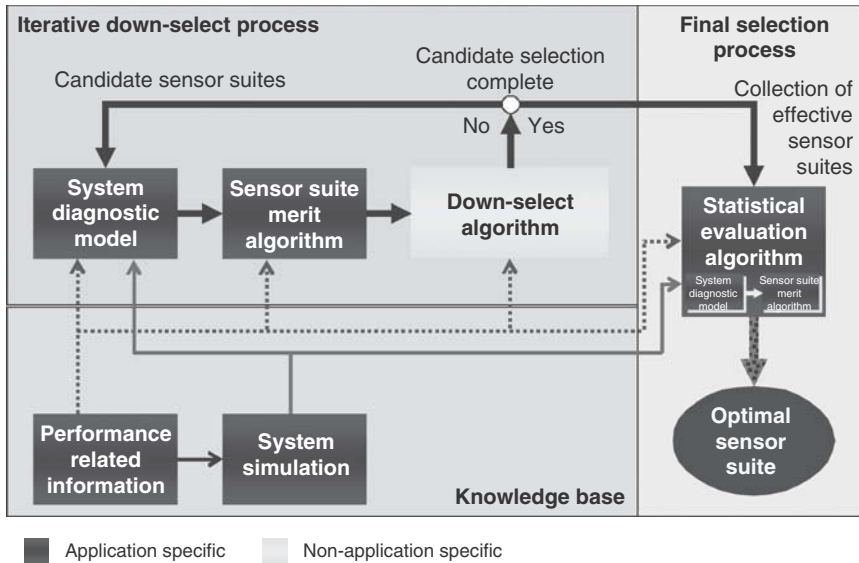


FIG. 8 iISHM Sensor Selection Strategy (S4) for optimizing sensor selection and placement.

quantitatively identifying the sensor compliment that optimally achieves the health assessment goals of a given system. Properly formulated, an S4 application can be used to determine whether or not existing sensors meet requirements for system health assessment, and, if not, to justify the addition of sensors that allow those requirements to be met. As shown in Fig. 8, S4 can be logically partitioned into three major elements: the *knowledge base*, the *iterative downselect process*, and the *final selection process*. The *knowledge base* consists of system design information and heritage experience together with a focus on components with health implications. The *iterative downselect process* identifies a group or groups of sensors that provide the highest fault detection and isolation performance for targeted fault scenarios. This process is further composed of three basic modules: the system diagnostic model, the sensor suite merit algorithm, and the downselect algorithm. The result of the *iterative downselect process* is a single sensor suite with the highest merit algorithm score (i.e., optimal) or a group of highest-performing (i.e., nearly optimal) sensor suites with closely matched merit algorithm scores. In the final selection process, the group of highest performing sensor suites is evaluated using a statistical algorithm that provides the final robustness test for each sensor suite. The result of the *final selection process* is a sensor suite that optimally achieves the system health assessment goals.

S4 has been used in studies to assess sensor coverage and diagnostic performance for a variety of aerospace propulsion systems. Further, a software

package [20, 21] was recently developed to help users implement their own application-specific versions of S4 and is available from the NASA Glenn Software Repository.

III. iISHM IN SYSTEMS DESIGN, INTEGRATION, AND ENGINEERING

Systems integration and engineering (SI&E) practices are employed to build complex systems. SI&E for aerospace systems has developed into its own discipline, although theories and concepts have not been formalized in an academic sense. NASA has published its formalized procedures [22] to standardize and promote the practice across the agency. The role of iISHM in SI&E is linked to the concept of iISHM-DMs, whereby every element that is part of a system comes with its own iISHM-DM that can be rolled up into an overall system iISHM-DM in a plug-and-play mode. In this sense, when two elements are assembled, the iISHM-DM of each element is incorporated into the iISHM-DM of the assembly. In this manner, DIaK compartmentalized in each element becomes immediately available to the iISHM-DM of the assembly. Figure 9 shows how systems integration is currently done, where knowledge and

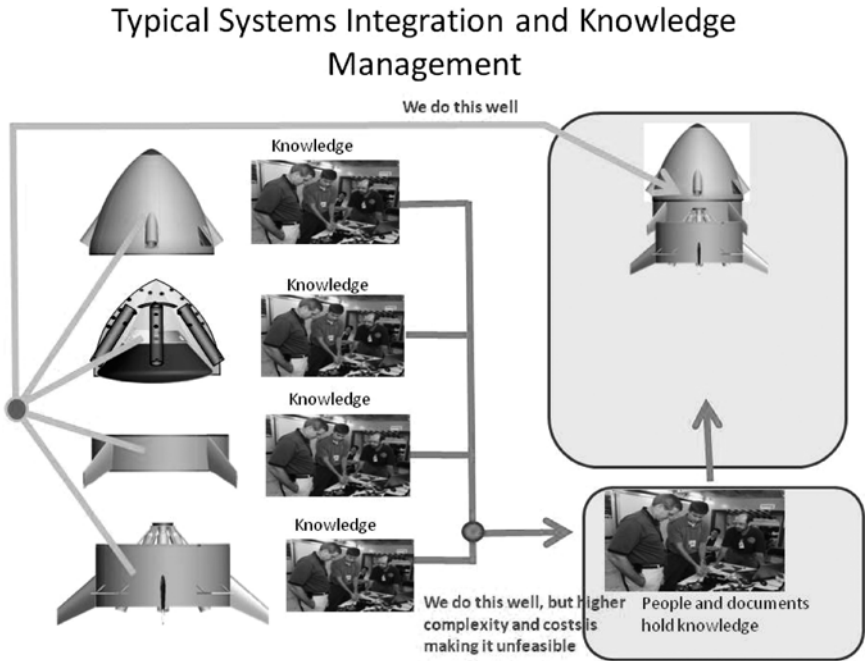


FIG. 9 Integration of subsystems today.

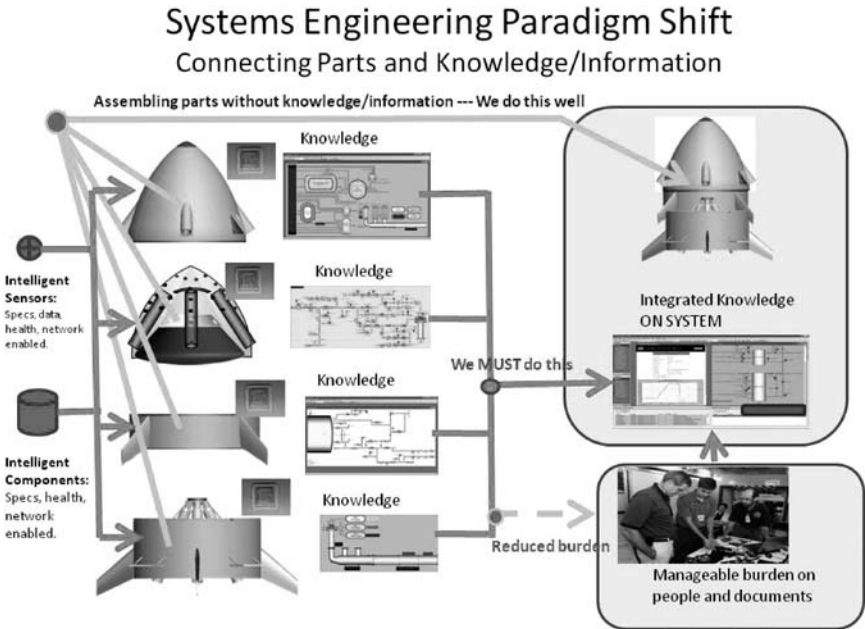


FIG. 10 Integration of subsystems with embedded ISHM.

information reside with people and documents. In contrast, Fig. 10 shows the concept of systems integration using systems with embedded knowledge (iISHM-DMs) providing comprehensive and continuous vigilance on the health of the elements throughout the integration process. This results in systems with embedded DIaK and the corresponding decrease of burden on people that operate the systems. The reason is because the iISHM-DMs provide immediate awareness about the design, integration, and condition of the assembly. Systems with embedded DIaK also decrease dependence on off-board documentation that is more difficult and costly to maintain and use.

The incorporation of iISHM-DMs as products of the design implies that parts of a system must be accompanied by DIaK relevant to determining the health of the parts. Failure modes and effects must be captured, as well as information such as expected life, specifications, usage, operational environments, etc. Specific advantages of integrating intelligent systems include the following: modular intelligent systems with advanced ISHM capability; faster and reliable integration, verification, validation, test, and mission readiness assessment; complete and continuous visibility of system condition throughout life cycle; decreased life-cycle costs; highly self-sufficient systems; and efficient evolution to future systems, as one builds upon integrated subsystems with embedded knowledge.

IV. INTELLIGENT CONTROL FOR iISHM-ENABLED SYSTEMS

Control of complex systems that are iISHM-enabled is a nascent area, simply because iISHM itself is also relatively new. The objective is for the control function to make use of system health information in order to achieve its objectives. Suspect (i.e., disqualified) sensors might be removed from use by critical control functions; anomalous components might need to be contained in order to maintain system function, and, in severe cases, new mission objectives might need to be identified. The paradigm implies that control systems become users of health information while making use of actuators to help further improve determination of the system health. This can lead to yet another area of control, specifically focused on helping the iISHM capability detect anomalies, diagnose causes, and determine effects. An example of a control system that incorporates sensor and actuator health information communicated using the IEEE 1451.1 Standard is described in [23].

Figure 11 shows a system diagram for a control strategy experiment that incorporates sensor and actuator health, where two key standards in the IEEE 1451 family of standards for smart sensors and actuators were used. The standards implemented include the transducer electronic data sheet (for automatic identification and specifications) and the NCAP (Network Capable Application Processor) used to communicate with the intelligent sensor. The control strategy selects algorithms depending on what faults sensors and/or actuators might exhibit.

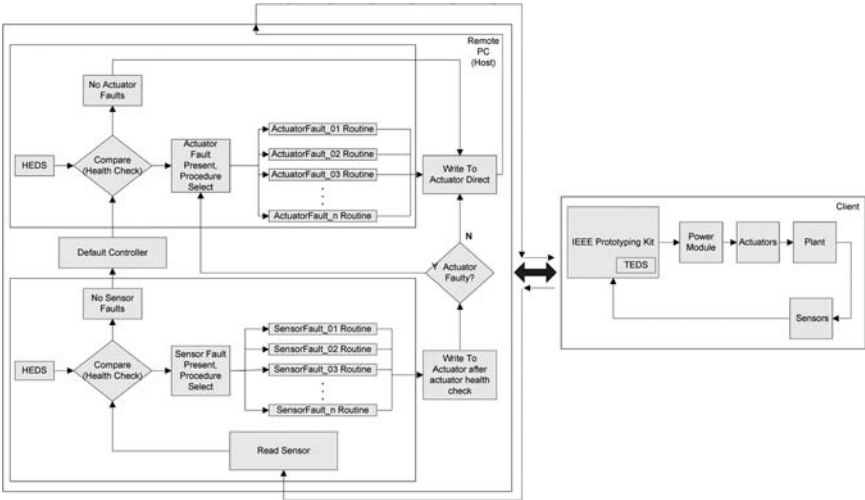


FIG. 11 Example control for health-enabled system [19].

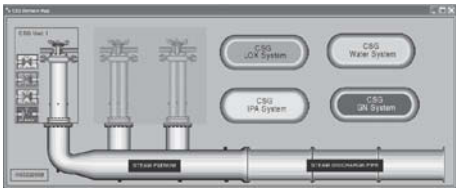
V. OPPORTUNITIES AND NEED FOR ADVANCES IN VERIFICATION AND VALIDATION

The chapter essentially describes iISHM capability implementation as purposeful management of DIaK with a focus on determining the health of each element in a system. The need to use knowledge, and hence inference engines and the complexities of parallel processing and reconciliation of potentially inconsistent outcomes that lead to anomaly determination, requires advances in verification and validation of the iISHM capability itself. This chapter only raises this issue, but the scope of this topic is broad; it is generic to knowledge-based systems, and it is left to be addressed by other colleagues.

VI. IMPLEMENTATION EXAMPLE: ROCKET-ENGINE TEST FACILITY AND TEST ARTICLE

A core pilot iISHM capability implementation was done for a rocket-engine test stand and its test article (chemical steam generator) at NASA Stennis Space Center [12]. Multiple objectives were achieved that incorporate many of the technologies, tools, and capabilities discussed in this chapter, but the most significant outcome was to achieve an implementation on an operational system and use it in real time, during operations.

The implementation embodied a physical system configuration that is shown in Fig. 7. Intelligent sensor functionality was achieved using the Virtual Intelligent Sensor Environment (VISE). This environment is able to process real-time data streamed from the data bus in the Test Control Center (TCC), or use historical data from files. In this case, real time means processing approximately 300 sensors and/or signals streaming at a rate of 250 samples per second. The VISE transforms every sensor and signal from the facility into its “intelligent” version. That is, it includes a TEDS for each sensor/signal, processes data streams to capture anomaly indicators, checks for exceedances of specification



iISHM-DM of CSG and test facility systems



Image of CSG system installed in the test stand

FIG. 12 iISHM-DM and image of system [12].

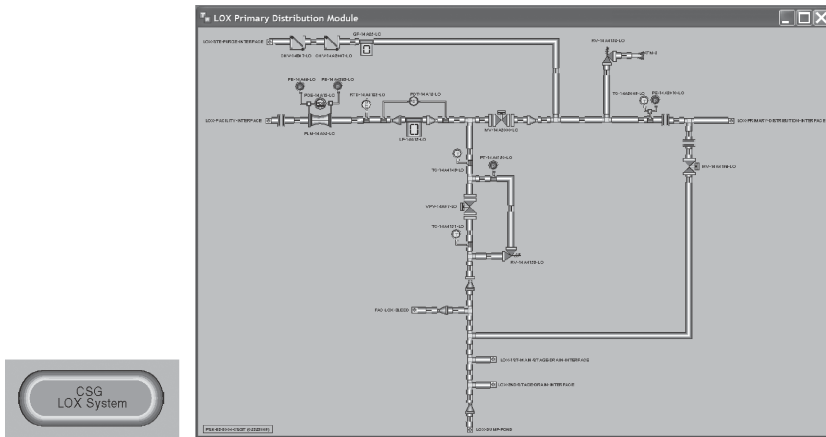


FIG. 13 Detail iISHM-DM shown by clicking on the “CSG LOX System” object from the top level diagram [12].

limits, and streams data to a bus using the IEEE 1451.1 standard for communications by smart sensors and actuators or NCAP. Data and sensor health information are also stored in the Health Assessment Database System (HADS) for analysis off-line. Intelligent sensor information (measurement plus health variables) are consumed by the iISHM-DM of the system, running in the ISHM (G2) environment. The VISE's software architecture is modular and can systematically be configured to accommodate more sensors and incorporate additional processes that operate on the measurement streams.

The ISHM-DM was developed using a toolkit built jointly by NASA and General Atomics (San Diego, CA) in the G2 environment (data available online at <http://www.gensym.com>). A description of the General Atomics version is provided in [13]. The DM includes subsystems of the test stand (ovals), subsystems that feed the test article (shown on left of CSG Unit 1), and the test article itself. Figure 12 shows a top view of the iISHM-DM (left), and an image of the system (right). Each entity in the iISHM-DM represents an object, and every system is a collection of interconnected objects, derived directly from schematic diagrams. Figure 13 shows details of the CSG LOX System displaying interconnected objects directly translated from the system schematic. Each element (pipe, elbow, valve, sensor, etc.) is an “intelligent” object that incorporates information describing who it is (ID and TEDS), what it is (class of object . . . such as a valve), and what it can do (parameters relevant to process models where the objects partake, for example operational limits from TEDS or component specifications, or potential failure modes). Figure 14 shows a top-level DM window along with subwindows with lists of objects, sensor TEDS, redline/blueline warning and occurrence lists.

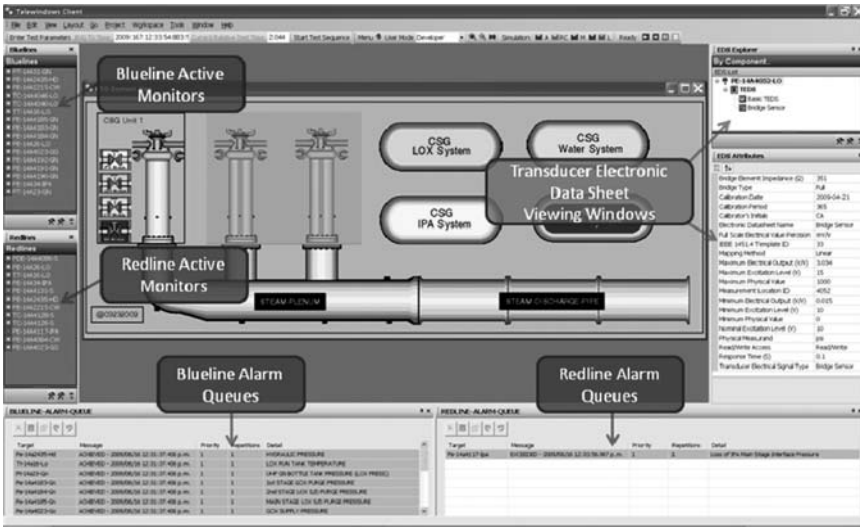


FIG. 14 An interface screen showing a top view of the ISHM-DM, along with windows containing information regarding TEDS and alarms [12].

When creating an iISHM-DM, the software automatically generates a knowledge base (KB). Objects are selected from a library in the toolkit, dropped in a workspace, and connected to reflect the schematic diagram. The KB generates configuration information derived from interconnections made at the moment objects are created and connected. These connection relationships are available for reasoning that might be done with multiple tools typical of object-oriented environments with an inference engine: procedures, methods, rules, and root-cause trees. Class membership relationships are also inherent in the object classes; for example, a temperature sensor is a member of the parent sensor class. Figure 15 shows a generic object class architecture suitable for creating iISHM-DMs. Multiple-inheritance can enable incorporating various categories of information for each object, for example, specifications, process models associations, principles of operation, etc.

An example that illustrates forcefully the need for a knowledge-based ISHM-DM is the implementation of a strategy to detect leaks in isolated subsystems. This strategy is a commonsense method of checking for leaks by operators. The condition for the leak check is to identify isolated subsystems that, by definition, should maintain pressure levels. Then identify pressure sensors in the subsystem, and check if pressure is maintained. If not, then the subsystem is leaking. That means that, initially, all member elements of the subsystem are suspect of leak, creating an ambiguity group. Additional information is used to reduce the ambiguity to a minimum number of suspect or confirmed leak sources.

Figure 16 shows the steps as implemented within the iISHM-DM. At system initialization, a procedure searches (navigates) through pipe objects, noting valves or other isolation elements. Note that all of the necessary information for this process is part of the class object definitions in the iISHM-DM (valve states, connection relationships, sources of fluids, etc.). When all isolated subsystems (IsoSubs) are identified, then, for each IsoSub, the procedure checks if any pressure sensors exist. This means that the procedure does not require that special sensors be installed; instead, it works with what is there. If an IsoSub does not have any pressure sensors, the procedure does not draw any conclusions about that IsoSub. For IsoSubs with pressure sensors, the procedure checks if pressure is maintained. Note that other reasoning can be applied as well to decide on whether the sensor measurements can be trusted, and to what degree. That assessment is being done through core procedures that take into account information from intelligent sensors and consistency checking through process models where the sensors are involved. In any case, if pressure is maintained, the procedure enters a monitoring/checking loop pertaining to any valve (or isolation element) configuration changes. If any occur, then only those IsoSubs affected by the valve with changed configuration are analyzed to determine deleted and newly formed IsoSubs. After initialization, the loop runs on a time schedule, but adjusting the IsoSubs occurs every time a valve (or isolation object) changes configuration.

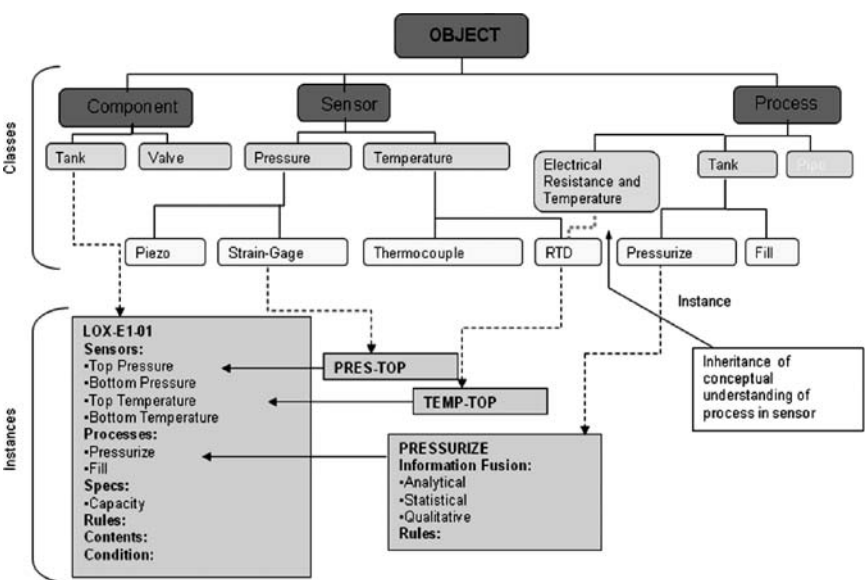


FIG. 15 Example object class definitions for iISHM-DMs.

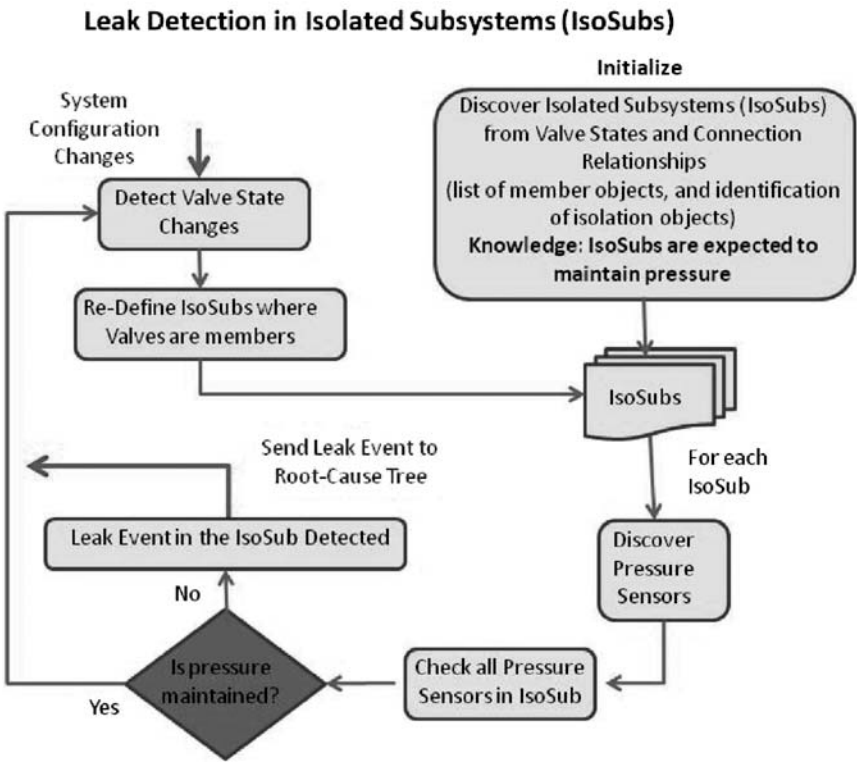


FIG. 16 Functional diagram for leak detection in isolated subsystems.

When the procedure detects a leak event in the IsoSub, it is sent to the leak for IsoSubs node of the generic root-cause tree and instantiated for the particular members of the leaking IsoSub. Figure 17 shows a generic tree for determining causes and effects of leak. The diagram constitutes the code itself, which is very

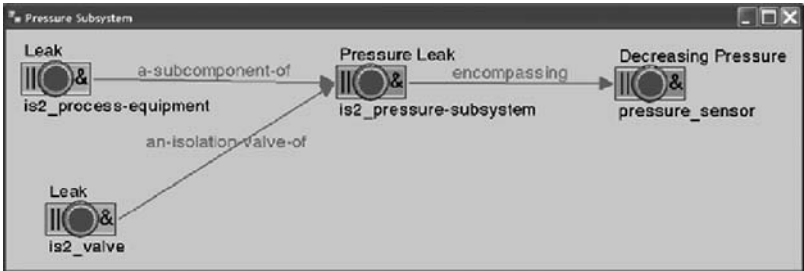


FIG. 17 Root-cause tree for diagnosis of leak detection in isolated subsystems.

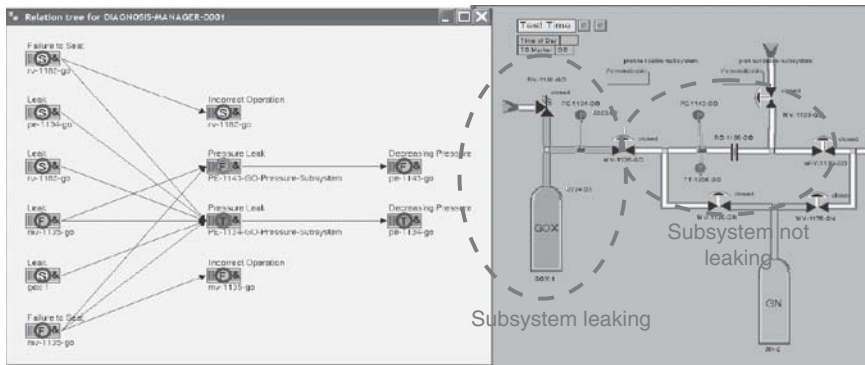


FIG. 18 Example instantiation of a root-cause tree for a leak event in an isolated subsystem.

expressive, drawing from information embedded in objects in the iISHM-DM. The tree indicates that a leak event in any is2_flow-subsystem (what has been called an IsoSub) can be caused by leaks on any is2_process-equipment (any element) that is a subcomponent of the IsoSub, but also treats separately a subcomponent that is an isolation valve. The reason is that isolation valves belong to adjacent IsoSubs, and if the adjacent IsoSub is not leaking, then one can immediately conclude that the valve is not leaking, thus reducing the size of the ambiguity group that may be causing the leak.

Figure 18 shows the occurrence of a leak event in an IsoSub instance. The right side of the figure shows an iISHM-DM diagram with gaseous oxygen (GOX) and gaseous nitrogen (GN) subsystems of a simple experimental rocket engine test stand. The leak detection procedure has identified all IsoSubs, including the two marked within the oval spaces. Pressure decrease in the sensor on the left has created a leak event in that subsystem. The event has been sent to the generic root-cause for leak (Fig. 17) and instantiated for that subsystem. When that happens, a detailed cause-effect diagram involving all elements in the subsystem is automatically created. Nodes for each subcomponent of the leaking IsoSub are overlaid with indicators that describe them as suspect (S), false (F), true (T), and also if the indication is inferred or direct. Note that the node corresponding to the valve isolating the two IsoSubs has an F indicator, meaning that it is not leaking. The other node for the valve also indicates that it is false that the valve has an anomaly called “failure to seat.” This conclusion comes from other root-cause trees describing failure modes of valves.

VII. CONCLUSION

This chapter describes concepts, architectures, paradigms, tools, and implementations of iISHM capability. The purpose is to show that iISHM capability must

be implemented as a knowledge-based capability, through management of data, information, and knowledge (DIaK), whereby “management” implies storage, distribution, sharing, maintenance, processing, reasoning, and presentation. The emphasis is also to note that iISHM capability increments “intelligence” of the system where it is implemented (the system knows the condition of every element). We can then talk about iISHM-enabled systems, with a potential to generate significant advances in systems design, integration, and engineering, as well as in systems control. The reader should also infer that much work is needed in developing iISHM-DMs, tools to create and use the iISHM-DMs, and implementation of standards for management of DIaK to achieve plug-and-play and interoperability. The concept of iISHM-DM encompasses an “integrating knowledge-model” about the entire system, specially incorporating knowledge that makes possible automating analysis of interactions across system elements and use in reasoning and decision making. Last, but not least, it is important to note that the iISHM DIaK architecture (DIaKA) described addresses the need for focusing on processes that take place in systems for consistency checking, leading to anomaly detection.

ACKNOWLEDGMENTS

The authors would like to thank NASA for providing the opportunity to work on advancing the area of Integrated Systems Health Management (ISHM). The authors also express their profound appreciation to the many individuals that through discussions and interactions have enriched their understanding of ISHM and made possible this chapter. Special thanks to Randy Holland, who believed in the need to advance this area, and provided initial funding and organizational support. Also thanks to John Schmalzel, Mark Turowski, Jon Morris, Richard Franzl, Mark Walker, and Meera Venkatesh for enlightening discussions, implementations, and commitment to advancing the ISHM area.

REFERENCES

- [1] Becerra-Fernandez, I., Gonzalez, A., and Sabherwal, R. *Knowledge Management: Challenges, Solutions and Technologies*, Prentice–Hall, Upper Saddle River, NJ, 2004.
- [2] Chen, Z., *Computational Intelligence for Decision Support*, CRC Press, Boca Raton, FL, 2000.
- [3] Surko, P., and Zakrajsek, J. F., “PTDS: Space Shuttle Main Engine Post Test Diagnostic Expert System for Turbopump Condition Monitoring,” Society of Automotive Engineers, SAE Technical Paper Series 922059, Oct. 1992.
- [4] Ramohalli, G., “The Honeywell On-Board Diagnostic and Maintenance System for the Boeing 777,” *Proceedings of the 11th IEEE/AIAA Digital Avionics Systems Conference*, 1992, pp. 485–490.

- [5] Davidson, M., and Stephens, J., "Advanced Health Management System for the Space Shuttle Main Engine," *Proceedings of the 40th AIEE/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA, Reston, VA; also AIAA Paper 2004-3912, 2004.
- [6] Chicatelli, A.K., Maul, W.A., and Fulton, C.E., "Propulsion IVHM Technology Experiment (PITEX)," NASA CR-2006-214238, July 2006.
- [7] Lee, K. B., "Smart Transducer Interface Standards for Condition Monitoring and Control of Machines," *Condition Monitoring and Control for Intelligent Manufacturing*, edited by L. Wang, and R. Gao, Springer Series in Advanced Manufacturing, Springer -Verlag, Berlin, 2006, pp. 347–372.
- [8] Lee, K. B., "A Standard in Support of Smart Transducer Networking," Inst. of Electrical and Electronics Engineers, Paper 1451, Jan. 2000.
- [9] Figueroa, F., and Melcher, K., "Integrated Systems Health Management for Intelligent Systems," AIAA Paper, 2011-1492, 2011.
- [10] Figueroa, F., Schmalzal, J., Walker, M., Venkatesh, M., and Kapadia, R., "Integrated System Health Management: Foundational Concepts, Approach, and Implementation," AIAA Paper 2009-1915, April 2009.
- [11] Figueroa, F., and Schmalzal, J., "Rocket Testing and Integrated System Health Management," *Condition Monitoring and Control for Intelligent Manufacturing*, edited by L. Wang, and R. Gao, Springer Series in Advanced Manufacturing, Springer-Verlag, Berlin, 2006, pp. 373–392.
- [12] Figueroa, F., Schmalzel, J., Morris, J., Turowski, M., and Franzl, R., "Integrated System Health Management: Pilot Operational Implementation in a Rocket Engine Test Stand," AIAA Paper 2010-3454, April 2010.
- [13] Walker, M., Kapadia, R., and Morris, J., "Integrated Design of On-Line Health and Prognostics Management," *Proceedings, Annual Conference of the Prognostics and Health Management Society*, 2009, http://www.phmsociety.org/sites/phmsociety.org/files/phm_submission/2009/phmc_09_30.pdf.
- [14] Kapadia, R., and Walker, M., "HealthMAP – A Model-Based Framework for On-Line Prognostics and Health Management (PHM)," AIAA Paper 2010-3500, 2010.
- [15] Sierhuis, M., Clancey, W. J., and van Hoof, R. J. J., "Brahms: a Multi-Agent Modeling Environment for Simulating Work Processes and Practices," *International Journal of Simulation and Process Modeling*, Vol. 3, No. 3, 2007, pp. 134–152.
- [16] Bajwa, A., and Sweet, A., "The Livingstone Model of a Main Propulsion System," IEEE Aerospace Conference, IEEE Paper 1444, Vol. 2-869, 2003.
- [17] Santi, L. M., Sowers, T. S., and Aguilar, R. B., "Optimal Sensor Selection for Health Monitoring Systems," AIAA Paper 2005-4485, July 2005.
- [18] Aguilar, R., Luu, C., Santi, L. M., and Sowers, T. S., "Real-Time Simulation for Verification and Validation of Diagnostic and Prognostic Algorithms," AIAA Paper 2005-3717, July 2005.
- [19] Sowers, T. S., Kopasakis, G., and Simon, D. L., "Application of the Systematic Sensor Selection Strategy for Turbofan Engine Diagnostics," GT-2008-50525, NASA/TM-2008-215200, June 2008.
- [20] Melcher, K. J., and Sowers, T. S., "Systematic Sensor Selection Strategy: Development and Implementation Software Tool," AIAA Paper 2011-1617, March 2011.

- [21] Sowers, T. S., "Systematic Sensor Selection System (S4) User Guide," NASA CR-2012/215242, Feb. 2012.
- [22] NASA Procedural Requirement 7123.1A NASA Systems Engineering Processes and Requirements w/Change, Nov. 4, 2009, http://nodis3.gsfc.nasa.gov/displayDir.cfm?Internal_ID=N_PR_7123_001A_&page_name=Chapter1.
- [23] Jethwa, D., Selmic, R. R., and Figueroa, F., "Real-Time Implementation of Intelligent Actuator Control with a Transducer Health Monitoring Capability," *International Journal of Factory Automation, Robotics, and Soft Computing*, No. 1, Jan. 2009, pp. 5–10.

Intelligent Propulsion Control and Health Management

Sanjay Garg*

NASA John H. Glenn Research Center at Lewis Field, Cleveland, Ohio 44135

I. INTRODUCTION

The modern dual-spool turbofan engine used for commercial and military aircraft has evolved into its current form over the past 60 plus years, starting with the first U.S. jet engine, which was built in 1942 by General Electric based on the British Whittle turbojet engine. Controls technology has played a critical role in advancing the performance, reliability, operating life, and safety of modern aircraft engine. Reference [1] provides a broad historical overview of the propulsion control technology development in the United States. The starting point of this chapter is the modern engine control referred to as full authority digital engine control (FADEC). The term “full authority” implies that the pilot has the full authority to move the throttle to any position during any operating phase of the airplane and the control logic in the FADEC ensures that the engine will be able to operate safely. The term comes from the early turbine engine control where the mechanical, electromechanical, and early electronic engine control limited the pilot authority for throttle movement based on the operating point within the flight envelope. This was to protect the engine from surging—a phenomenon during which the airflow in the compressor reverses, causing severe damage to the engine. Today’s modern aircraft engines are typically equipped with dual-channel FADECs with extensive built-in test functions for the control electronics, some basic form of embedded engine analytical models to perform sensor and actuator validation checks, and algorithms to track life usage.

This chapter will focus on the control logic that is incorporated in the FADEC to provide safe and reliable operation of the engine throughout the operating envelope while also guaranteeing a certain operating life of the engine. Furthermore, this chapter focuses on the high-bypass turbofans used in commercial aircraft. Typically an engine is certified to provide a minimum level of performance, that is, thrust greater than a specified value for a given throttle setting at a given operating condition (ambient pressure and temperature), and a specified

*Chief, Controls and Dynamics Branch; Sanjay.garg@nasa.gov.

maximum thrust rise time for throttle movement from idle to maximum thrust setting. For commercial aircraft, an engine operating cycle consists of throttling the engine from idle to max thrust for takeoff, setting the desired thrust levels for climb and cruise, and then setting the engine to idle for landing. As will be discussed later in this chapter, the control logic plays a critical role in meeting these performance guarantees while ensuring safe operation of the engine. In the following, a brief description covering the operation of a turbofan engine, operational and safety limits, and mathematical modeling of the engine is provided first to help lay the groundwork for main body of the chapter, which focuses on the control of the turbofan engine. After the basic introduction to engine operation and dynamic modeling, the chapter is organized into three main sections: state-of-the-art of engine control, some retrofit-concepts for increased life and reliability, and model-based control and diagnostics. The term “retrofit-concept” as used in this chapter implies minor changes to be implemented in the existing control architecture to achieve the desired objectives. The term “model-based” implies that there is an onboard model of the engine that adapts to changing engine condition and provides estimates of quantities of interest that are either not measured or are not physically measurable.

II. TURBOFAN ENGINE OVERVIEW

The intent of this brief overview of turbofan engine operation is to provide the reader some context of the challenges associated with engine control. References [2–4] will enable the reader to gain a more complete understanding of this background information. The discussion in this section is based on the material in [3].

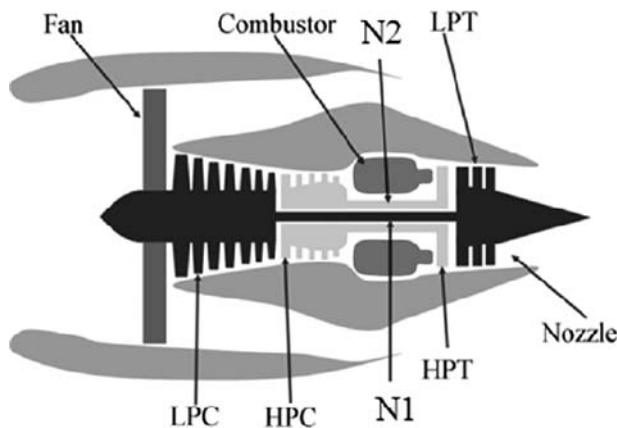


FIG. 1 Schematic of a high-bypass twin-spool turbofan engine.

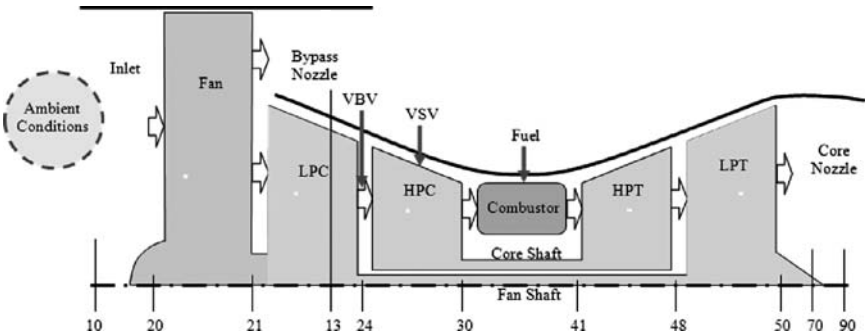


FIG. 2 Diagram of the typical engine components and station designations.

Figure 1 shows a schematic of a modern high-bypass dual-spool commercial aircraft turbofan engine. The air supplied by the inlet to the engine is compressed by the fan. A major portion of this compressed air is ducted to the outer gas path flow, referred to as “bypass,” while the rest is routed to the engine core. The core air is further compressed by two multistage compressors, LPC (low-pressure compressor) and HPC (high-pressure compressor). The compressed air is mixed with fuel and burned in the combustor. The two turbines, HPT (high-pressure turbine) and LPT (low-pressure turbine), extract work from the hot combustor air. The LPT drives the LPC and the fan while the HPT drives the HPC. The core air is exhausted through the nozzle. In this engine configuration, the bypass air exhausted over the engine core creates most of the thrust. The bypass ratio is the ratio of the fan compressed air going through the bypass duct to the core air. The turbofan engine achieves fuel-efficient operation by accelerating a large amount of air through a small velocity change. In Fig. 1, N1 and N2 refer to the speeds of the two spools with N1 being fan speed and N2 being the core speed. Note that in general N1 is much lower than N2.

A standard notation is used to describe the various points along the engine components. This notation is shown in the diagram of Fig. 2. Note that although fuel flow is the main control variable to set the engine operation for a desired level of thrust, typical modern engines have other control variables such as a variable bleed valve (VBV) and variable stator vanes (VSV) to allow for safe and efficient operation of the engine. The VBV is to prevent the LPC from entering stall at any operating condition by providing pressure relief at the LPC exit. The VSV are typically located at the forward stages of the HPC and are used as a means to maintain compressor stability and prevent choking of the downstream compressor stages. These other control variables are typically scheduled as a function of the engine operating condition and are not used for direct feedback control.

Because the turbofan engine is a thermomechanical device, there are various limitations on its operation. Some of the limits of concern from a control perspective are structural and operational limits. To maintain the structural integrity of

the rotating components, there is a maximum limit on the allowable fan and core shaft speeds. Also, because the blades in the first stator stage of the HPT see the highest flow temperatures (at the exit of the combustor), the maximum temperature at station 41 (T41) has to be limited to prevent the blades from getting hot enough to lose structural integrity. Operational limits consist of ensuring that adequate stall margins are maintained for the LPC and HPC throughout the engine operating envelope to prevent the compressors from approaching stall during engine transients and ensuring that a minimum fuel flow is maintained at any operating condition to prevent combustor blowout. From an acceptable operating life perspective, the maximum turbine inlet temperature (T41) is further restricted to prevent excessive thermomechanical fatigue damage to the HPT and LPT components. The objective of the control logic design is to provide acceptable performance while ensuring that the engine operation remains within the structural and operational limits.

Dynamic engine models are developed to design and evaluate engine control logic. The following provides a brief overview of the engine dynamic modeling. The interested reader can find details in [3] and the references therein. Steady-state engine performance is obtained from cycle calculations derived from component performance maps obtained through detailed component modeling and component tests. Corrected parameter techniques are used to reduce the number of points that need to be evaluated to estimate engine performance throughout the operating envelope. Dynamics are modeled through inertia (the rotor speeds), combustion delays, heat soak and sink modeling, etc. This is a computationally intensive process because it is important to maintain mass/momentum/energy balance through each component. A detailed thermodynamic cycle deck is developed using this procedure, and parameters within this model are adjusted based on experimental data to match the tested engine performance characteristics. Simplified models from this thermocycle deck are generated to develop and evaluate the control logic.

III. STATE OF THE ART OF ENGINE CONTROL

The basic objective of the engine control logic is to provide smooth, stable, and stall-free operation of the engine via a single pilot input—throttle or PLA (power lever angle) with no restrictions on throttle movement. The engine control should provide reliable and predictable throttle movement to thrust response throughout the engine operating life. Some of the challenges in engine control design are as follows:

- Thrust, the primary controlled variable of interest, cannot be measured.
- Changes in ambient condition and aircraft maneuvers cause distortion into the fan/compressor, and so the engine control has to be robust under these conditions.

- Parameters that need to be limited, such as turbine inlet temperature, minimum stall margin, etc., cannot be directly measured.
- Engine components degrade with usage, and so the control logic has to be robust to provide guaranteed performance in the presence of significant plant model deviations. The effect of this degradation on engine performance is modeled through a set of influence coefficients called health parameters [4, 5].

Additionally, the harsh operating environment of the engine, including high temperatures and large vibrations, places certain constraints on the control hardware. One of the constraints, which are important for control logic development, is the limitations it places on the number and type of sensors that can be placed on the engine. A typical engine control system has 6–7 sensors including the two rotor speeds (N1 and N2), pressure and temperature measurements at 2–3 engine locations such as the fan inlet, LPC and HPC exits, and an exhaust gas temperature (EGT) sensor that measures the gas temperature at the exit of the LPT. EGT, which increases with usage, is primarily used as an indication of the remaining useful engine life.

A typical modern engine control logic is shown in the block diagram of Fig. 3. Because thrust cannot be directly measured, a variable that can be sensed and is directly correlated with engine thrust is selected to be the primary controlled variable through the PLA. Figure 3 shows the engine fan speed (N1) as the primary controlled variable. Note that because the majority of the thrust in commercial turbofan engines is created from the fan bypass flow, N1 is an appropriate variable to regulate to achieve the desired throttle to thrust response. Another variable that can be used as the primary controlled variable is the engine pressure ratio (EPR) defined as the ratio of the pressure at the LPT exit to the pressure at the fan inlet.

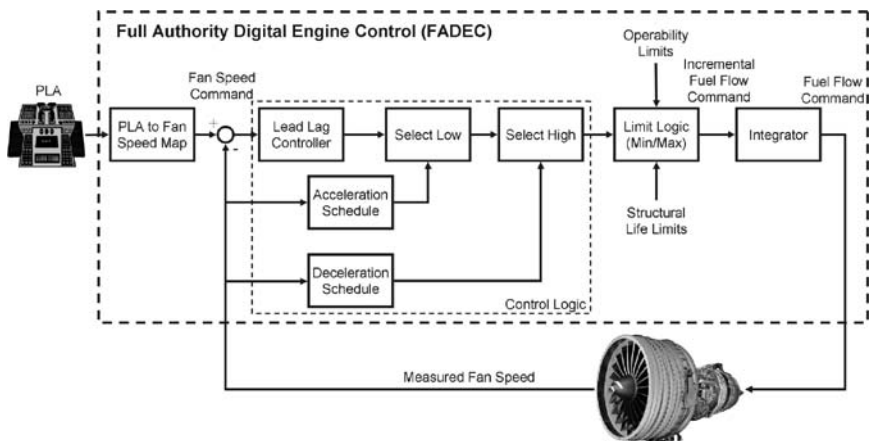


FIG. 3 Block diagram of typical modern engine control logic.

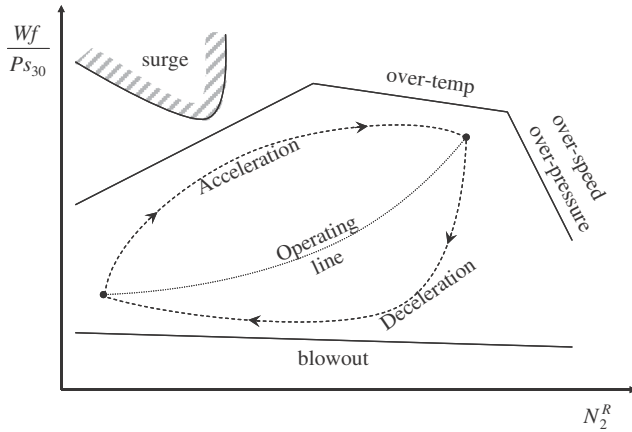


FIG. 4 Engine limit protection implementation philosophy.

In Fig. 3, a PLA setting creates a reference command for N_1 . A simple lead-lag compensator creates a fuel flow command based on the difference between the commanded and measured fan speed (note that an incremental fuel flow command is generated for the architecture shown in Fig. 3). However, this command cannot be implemented as is because the engine structural and operational limits, as discussed earlier, need to be taken into consideration. The acceleration and deceleration schedules are implemented primarily to prevent compressor stall during transients. Secondly, the acceleration limit is an additional measure to prevent overstress of the rotor, and the deceleration limit protects against combustor blowout. These schedules are a function of the corrected core engine speed (N_2^R) and impose a limit on the ratio of fuel flow to combustor inlet pressure (Wf/Ps_{30}) because this ratio, which can be computed from measurements, relates closely to surge and temperature margins. Figure 4 shows a typical example of how these schedules protect the engine operation from violating the structural and operational limits.

Because the acceleration schedule imposes a maximum limit on the fuel flow, the lower of the fuel flow imposed by the acceleration limit and that required to track the regulated variable is selected through the “select low” block in Fig. 3. The output of this block is compared with the minimum fuel flow command required to stay within the deceleration limit, and the higher value is selected through the “select high” block. The output of the “select high” block is further compared with the fuel flow required to stay within additional limits, such as the maximum core speed, etc., and the selected incremental fuel flow is processed through an integrator. This min-max selection control architecture ensures that the fuel flow command sent to the fuel flow actuation system is such that the engine operation will avoid exceeding any of the limits.

The engine acceleration and deceleration schedules are determined through simulation and analytical studies, and the engine set point schedules are selected such that the desired performance setting can be achieved without operating on a limit in the steady state. The performance variable regulation and other limit variable regulation is typically implemented through PI (proportional + integral) control with the control gains designed using linear models and scheduled typically as a function of PLA and Mach number and other additional variables that define the engine set point. The control design is evaluated throughout the operating envelope using a nonlinear engine simulation and for engines at various levels of condition in the operating life, as modeled through the variation of health parameters.

One of the typical transient test cases for engine control is “burst-chop” wherein the PLA is moved from idle to full (burst) and a few seconds later it is moved back to idle (chop). The “burst-chop” results from the nonlinear engine simulation evaluation for the control design discussed in [3] are shown in Figs. 5 and 6. In Fig. 5, TRA stands for throttle resolver angle, which is analogous to PLA discussed earlier, N_f and N_c are fan and core speeds, respectively, and T48

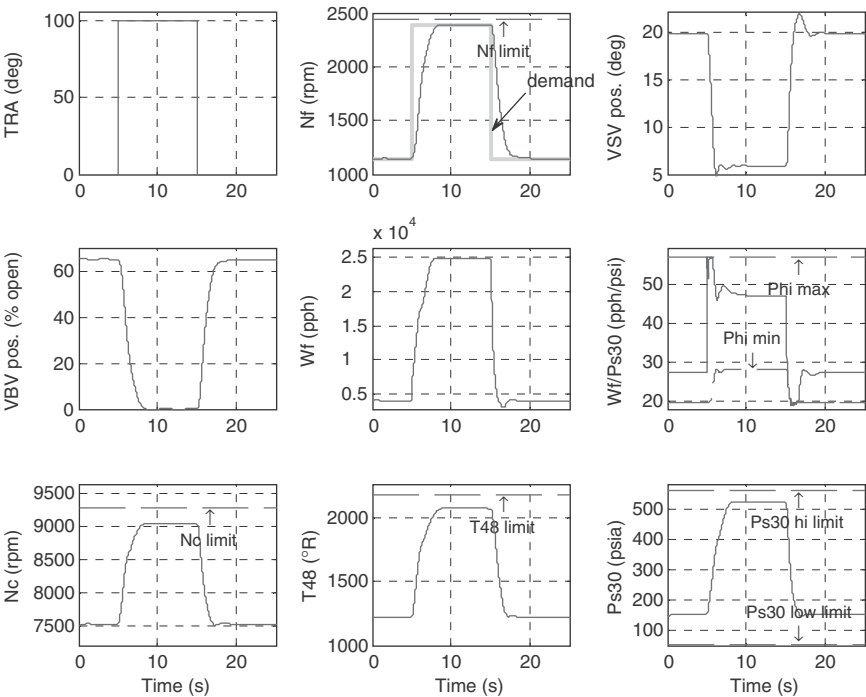


FIG. 5 Engine response for burst and chop (from [3]).

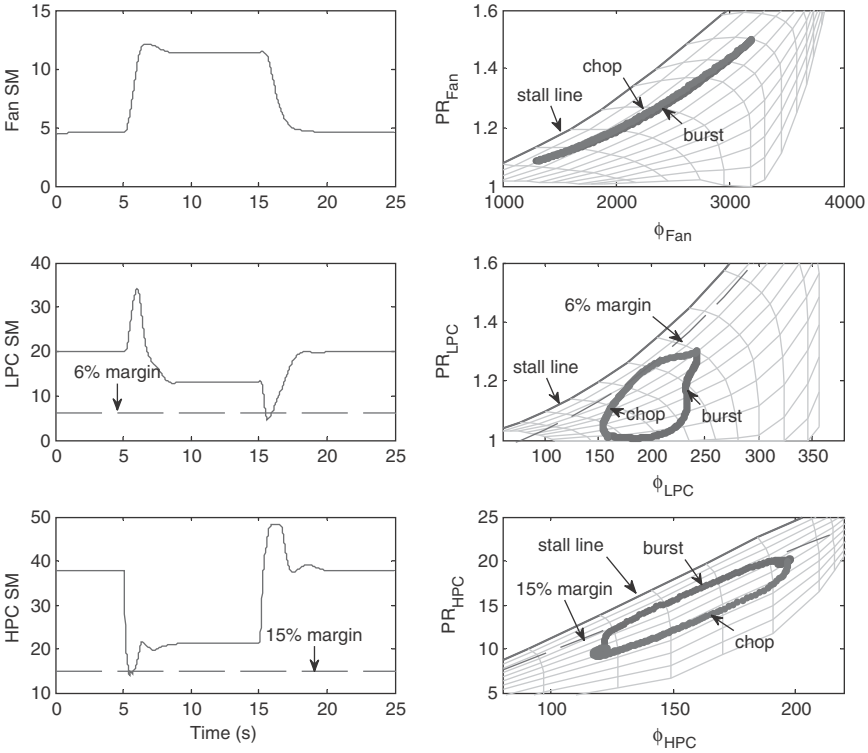


FIG. 6 Burst/chop trajectories shown in relation to the fan, LPC, and HPC pressure ratio vs corrected flow maps (from [3]).

is the exhaust gas temperature. The plots in Fig. 5 show that the fan-speed compensator (in control the majority of the time) produces a smooth fan speed response that does not violate any limits. The VBV and VSV actuators follow schedules that are inversely proportional to the corrected fan and core speeds, respectively. The only limiter that comes into play is the Wf/P_{s30} saturation limit; the maximum is encountered during burst and the minimum encountered during chop. Figure 6 shows the transient response of the stall margins (SM) in the fan and compressor components. The left plots show the stall margins as a function of time whereas the right plots show the stall margins as a function of the component performance map. The performance map is a plot of pressure ratio (PR) across the component as a function of the corrected mass flow Φ . Viewed on the pressure ratio map, the fan stall margin evolves within a narrow region and hence does not require limit protection. The LPC and HPC, on the other hand, have large-magnitude transient responses that require protection by the Wf/P_{s30} limit. By virtue of their design, the limits prevent the LPC stall margin from falling

below 6% during chop and the HPC stall margin from falling below 15% during burst.

Once the control design is determined to provide acceptable performance without violating any of the structural and operational limits, it is coded into software for implementation into the control hardware. The control gains are further adjusted based on engine ground and altitude tests and finally flight tests before releasing a product version of the control logic.

IV. SOME RETROFIT INTELLIGENT ENGINE CONTROL CONCEPTS

The engine control architecture described in the preceding section has been developed by the engine companies using their experience over a period of many years and has demonstrated the capability to meet the engine certification requirements in terms of guaranteed performance and safe operation throughout the engine operating life. There is a real hesitancy on the part of the engine companies to deviate from the min-max control selection architecture because of the extensive costs associated with developing, testing, implementing, and certifying a totally new control architecture. In this section, three enhancements to the existing engine control architecture are considered that have the capability to provide increased reliability, longer on-wing life, and improved operational safety, respectively. All of the three concepts considered can be implemented as a retrofit to the existing engine control logic with addition of either some new control blocks or modification to how the limit logic is developed and implemented.

A. HIGH-RELIABILITY ENGINE CONTROL

Typical modern turbofan engines have a dual hardware redundancy architecture in all of the control components including sensors. For control purposes, the sensor data have to be validated before processing through the control logic. Typically the sensor data validation scheme consists of reasonability range and rate limit checks. Through simulation and experimental studies, a range of values for the sensed variable is determined based on what will be expected throughout the engine operating envelope. If the sensed value is beyond this range, then that particular sensor is considered to be faulty. Similarly if the value from a sensor changes faster than an acceptable rate, then that sensor is considered to be faulty. In a dual hardware redundancy architecture, the two sensors for any variable will typically have some differences in their value because of noise, manufacturing tolerances and any aging factors. If this difference is below a certain threshold, and both the sensor readings are within their acceptable range, then the two values are averaged, and this average is used as input to the control logic. If the sensor validation approach shows one of the sensors to be failed, or if the two sensors provide significantly different values, then the engine control is reverted to a safe mode, and a warning is issued to the pilot to fly the airplane

to a location where a maintenance action can be taken. This is referred to as D0 maintenance action. This current approach to sensor validation results in a significant cost to the airlines.

Various approaches for improved sensor validation have been suggested in the literature. One such approach is to leverage the analytical redundancy inherent in the suite of sensors that are used for engine control. A high-reliability engine control (HREC) approach based on using an auto-associative neural network (AANN) was developed in [6]. This feed-forward network architecture has output data that reproduce the network input data (see Fig. 7). The AANN consists of three layers (not counting the input and output layers): the information compression layer and the information regeneration layer, which are interconnected through the bottleneck layer. The compression layer compresses the data into a reduced-order representation, eliminating redundancies and extracting the key features (principal components) in the data. Reducing the number of dimensions is the key characteristic of this architecture. The regeneration layer recovers the encoded information from the principal components. This way the AANN reproduces estimates of the sensed values, and because the principal components are obtained through analytical redundancy, the AANN output corresponding to a faulty sensor still provides a good estimate of the true value of the sensed variable. Fault detection logic identifies the unique sensor failure based on the pattern of the error vector of the input and output of the AANN.

The AANN approach was applied to the simulation of a commercial turbofan engine. The AANN was trained to identify sensor faults by comparing the outputs of the network with the sensor readings—the input to the network. Additionally, in case of a sensor failure, the estimated value from the trained AANN was used to maintain the desired control of the engine. Figure 8 shows an example result, from

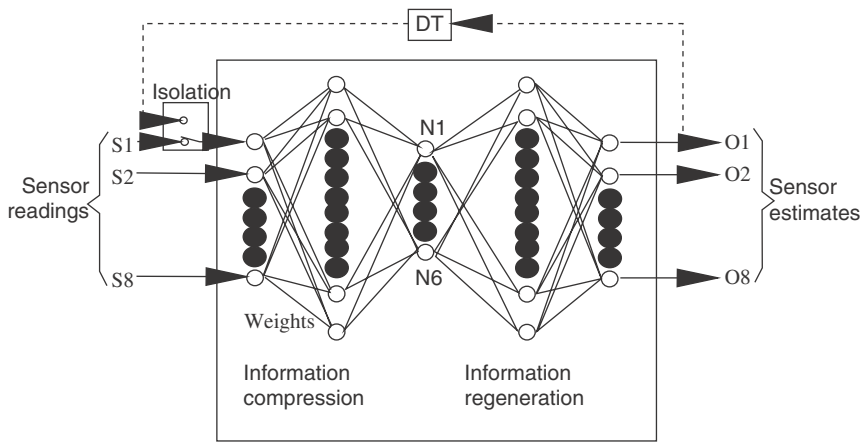


FIG. 7 Auto-associative neural network architecture.

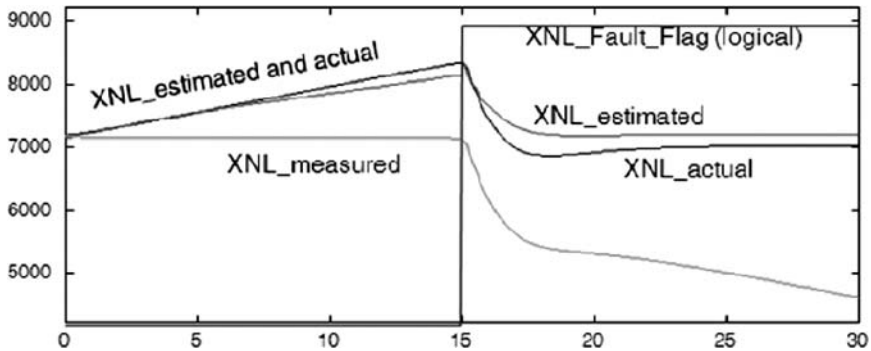


FIG. 8 High-reliability engine control result for a low-speed (xnl) sensor failure.

the simulation studies summarized in [6], for the case of a fan speed (XNL in Fig. 8) sensor failure. The controller operating on XNL_measured maintains speed (as measured) at 7000 rpm whereas the actual speed is drifting. The AANN estimate tracks the actual speed. Once the estimate is significantly different from the measured value (at 15 s), the failure accommodation logic activates and transfers the control to the estimated value, resulting in the desired speed being maintained. Such an approach can allow a sensor failure to be designated as a D10 condition, that is, the airplane will need to be flown to a maintenance facility within the next 10 days of operation. Such a designation can significantly reduce the overall cost of accommodating the sensor failure and result in increased reliability of operation.

B. LIFE EXTENDING CONTROL

With the recent emphasis on reducing engine operating cost, the industry is interested in developing technologies that will allow the engine and its components to operate (remain on wing) longer, thus increasing the time between engine overhauls. How the engine is controlled has a severe impact on the life of the components. Typically, the propulsion system control design engineer attempts to get the maximum performance out of the system while maintaining safe operation. Recent studies have shown that small changes in engine operating parameters, such as turbine inlet temperature, can have a significant impact on the damage accrued by engine components while having little to no effect on engine performance. NASA Glenn Research Center (GRC) developed the concept of life extending control where the engine control system is designed to achieve the desired performance while minimizing the damage accrued in engine components, hence maximizing the usable engine life. The feasibility of this concept was demonstrated for the space shuttle main engine through simulation. Efforts in collaboration with industry led to the development of intelligent

life extending control (ILEC) [7] with a focus on optimizing the engine acceleration schedule to minimize the damage accumulation in the hot gas path components while meeting the Federal Aviation Administration (FAA) thrust response rise time requirement for PLA command from idle to maximum power.

During takeoff, when the pilot pushes the throttle to move the engine from idle to maximum power, the engine control generates a fuel flow command based on acceleration logic that ensures that the maximum thrust is achieved within a time limit based on FAA requirements. The engine components accumulate damage during this transient due to the quick and large changes in temperature and aerodynamic loads. By adjusting the acceleration logic such that the time to achieve maximum thrust is just within the FAA requirements, the temperature and load changes on the engine components can be kept to the minimum required to meet the performance. This will result in reduced damage accumulation for each takeoff and hence increase the amount of time the engine can stay on the aircraft before a major overhaul is required. The idea of ILEC is then to design a smart acceleration logic for engine control that will minimize the thermomechanical fatigue damage accumulated during a typical engine acceleration transient from idle to full power without any noticeable loss in engine performance. A typical baseline acceleration schedule for a commercial turbofan engine, as well

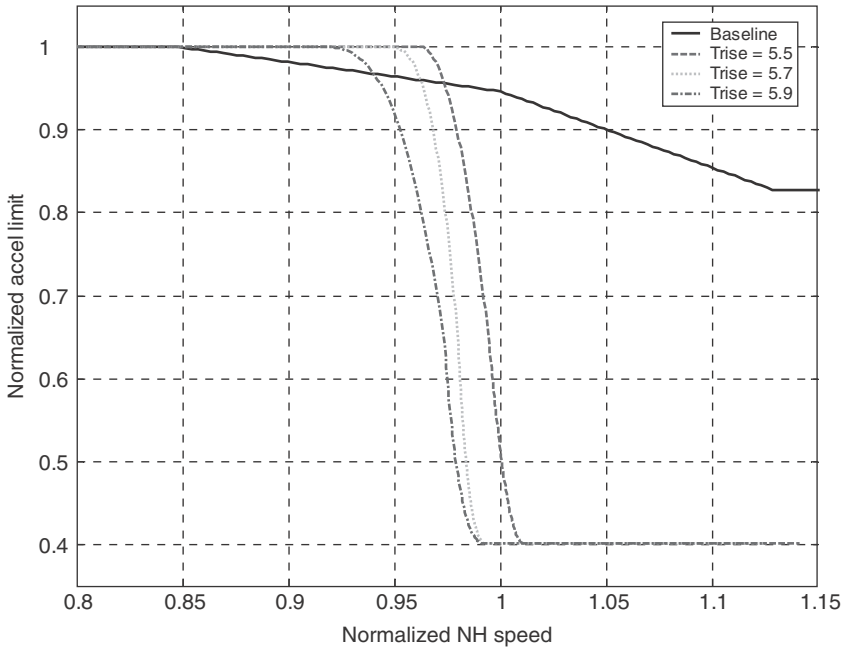


FIG. 9 Baseline and optimized acceleration schedules for takeoff acceleration process.

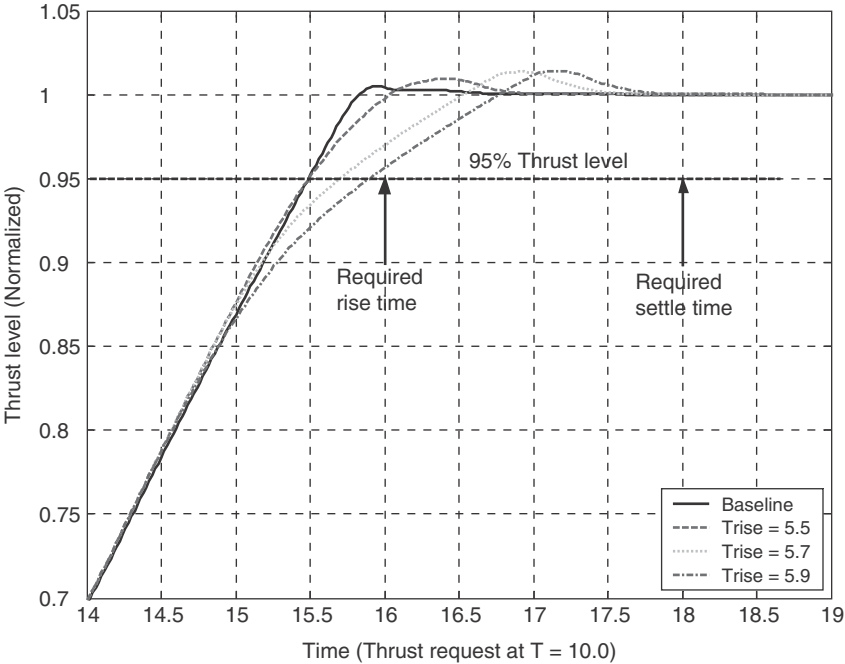


FIG. 10 Thrust response curves for the nominal and optimized acceleration schedules.

as schedules optimized to achieve acceptable takeoff transients, while minimizing life consumption [7] for a specific hot-section engine component, are shown in Fig. 9. The acceleration schedule is defined as the limit on the core speed rate of change as a function of the core speed (NH in Fig. 9). The thrust transients that correspond to these schedules are shown in Fig. 10. In Fig. 10, Trise is defined as the time for the thrust to reach 95% of the maximum power level. The optimized schedules were developed for Trise ranging from the value for the baseline schedule to the maximum allowable as per FAA certification requirements. Simulation results demonstrated that the optimized acceleration schedule decreased component life consumption during takeoff significantly for the same rise time, and extending the rise time slightly reduced the life consumption even more [7].

C. ENGINE PERFORMANCE DETERIORATION MITIGATING CONTROL

As an aircraft engine deteriorates with usage, there is noticeable change from the throttle setting to the thrust response. In a workshop sponsored by NASA to identify technology development needs for reducing pilot workload and increasing autonomy with respect to operation of aircraft engines, various pilots stated

that the asymmetric thrust, caused by this deteriorated engine response, causes additional workload for them in having to make adjustments to individual throttles in a multi-engine aircraft. As discussed earlier, because thrust is not measurable, typical engine control consists of tracking a fan speed command based on a throttle setting. The fan speed to thrust relationship varies with engines due to manufacturing tolerances and changes as the engines deteriorate with usage. For a multi-engine aircraft, this difference in fan speed to thrust relationship results in variations in throttle to thrust response for different engines. Aircraft accident analyses have also identified uncommanded asymmetric thrust to be one of the major causes of safety incidents related to the propulsion system.

One approach to addressing the issue of providing consistent throttle to thrust response is to change the overall control architecture to a model-based control as will be discussed later in this chapter. However, such a model-based control approach is expected to take a long time to reach a high enough level of technical maturity to be able to meet the stringent certification requirements for safe operation of aircraft engines. The FADEC used for implementing typical engine control has both throughput and processing limits, which make a full model-based control implementation very challenging. Additionally, the current engine control already contains limit logic that is designed to guarantee safe operation of the engine over a wide operating envelope and under varying atmospheric conditions and to enable an economically viable on-wing life. For these reasons, it is imperative to find a solution to the consistent throttle to thrust requirement, which can be implemented within existing FADEC capabilities, and will require minimal changes in the existing control implementations for operational safety.

An engine performance deterioration mitigating control (EPDMC) retrofit architecture that can alleviate the problem of asymmetric thrust due to uneven engine deterioration, potentially reducing workload and improving aircraft safety, was developed in [8]. The architecture is depicted in Fig. 11. It is built on the existing FADEC control logic but with an outer-loop thrust control that

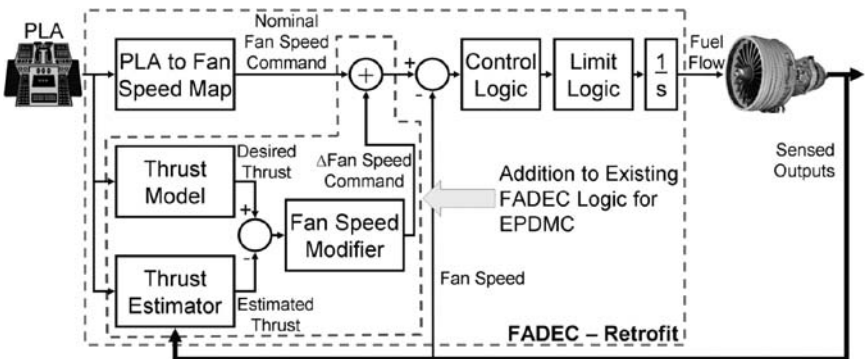


FIG. 11 Engine performance deterioration mitigating control architecture.

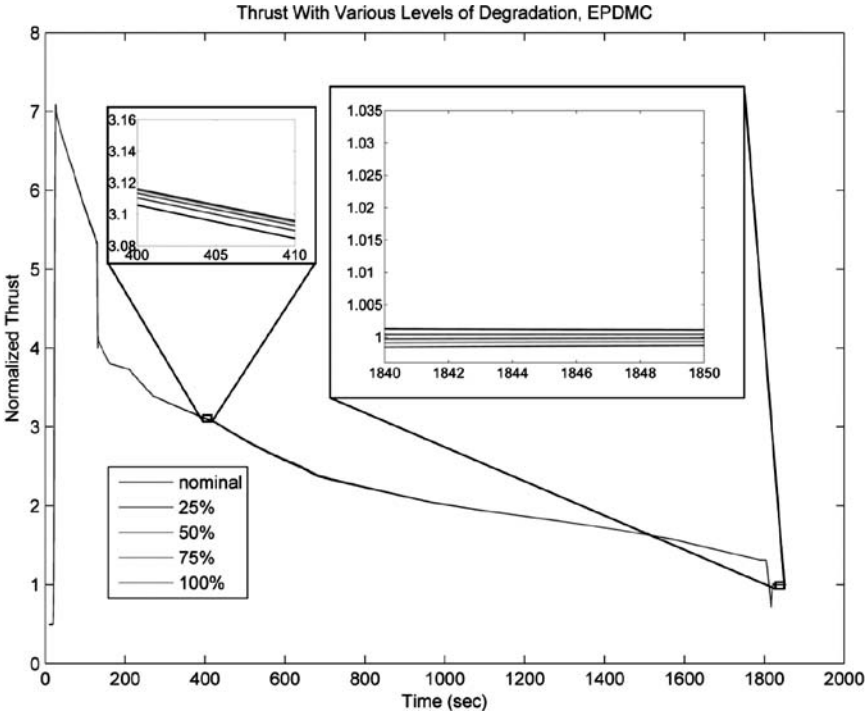


FIG. 12 Normalized thrust through takeoff/climb/cruise transient scaled by nominal level at cruise, with EPDMC retrofit. The insets show the nominal thrust level is essentially maintained regardless of engine deterioration.

gives a consistent throttle to thrust response. Here the inner-loop control is simply the existing FADEC logic with fan speed feedback. This retains all of the certified control and limit logic that protects the engine. Three blocks have been added to facilitate the adjustment of the fan speed command to effectively control thrust. The main elements of EPDMC outer-loop control are 1) a thrust estimator that provides an accurate estimate of the engine thrust based on available sensor measurements and actuator commands; 2) thrust demand logic that contains a thrust model of a “nominal” engine for a given throttle setting; and 3) a PI control, which provides an incremental fan speed command to the FADEC to compensate for the difference between estimated thrust and thrust demand.

The EPDMC approach has been applied to an engine simulation representative of a modern commercial aircraft engine and has been shown to maintain throttle to thrust response in the presence of engine degradation with usage. Figure 12 shows the normalized thrust (normalized with the cruise thrust value) through a typical takeoff/climb/cruise trajectory with the EPDMC active

for various levels of engine degradation—from new to fully degraded. As seen from this figure, the throttle to thrust response is maintained in a very tight bound throughout the varying engine condition. As documented in [8], with the baseline fan speed control, the simulation results showed a variation in thrust of up to 3% of nominal cruise thrust as the engine degraded with use. Note that EPDMC architecture is suggested for implementation only to account for the thrust asymmetry due to engine degradation differences in a multi-engine aircraft. Additional research still needs to be done to identify the cause for any observed uncommanded engine thrust asymmetry. If the underlying cause is not just deterioration, such as faulty throttle linkages, which can result in much larger thrust asymmetries, continuing to implement EPDMC might mask the problem and worsen the safety situation.

V. MODEL-BASED CONTROL AND DIAGNOSTICS

The discussion in this section is based on the NATO (North Atlantic Treaty Organization) RTO (Research and Technology Organization) report on “More Intelligent Engines” [9].

So far we have focused primarily on the engine control challenges without directly taking into consideration any information about the condition or health of the engine. Engine health monitoring can provide valuable information to enhance the engine control performance. The various on-wing health monitoring systems of today, which are a collection of separate, unrelated technologies, provide a basic level of monitoring. Their capabilities are relatively limited, and the information they provide is used mostly to initiate maintenance actions, not for real-time decision-making. One instance in which the information is used on wing is for sensor validation. The controller has some simple logic to perform basic limit or rate of change checks on engine sensors and actuators. In some cases, onboard engine models are used in conjunction with the controller’s own sensor voting scheme to help determine which sensor is correct when redundant sensors disagree. Current engine vibration monitoring systems sample at a relatively low frequency—too low to capture much significant or useful information on the vibratory modes of the system. They check the vibration magnitude to determine that it is within a normal range. Magnitudes that are too high might indicate a bearing failure or engine imbalance; magnitudes that are too low might indicate a faulty sensor or seized engine. Lubrication system monitoring is performed using a magnetic chip detector to determine the existence of ferrous debris in oil; this is an indication of part wear. Life-cycle counts are performed on wing. Engine parts, especially those in the hot section, may experience a maximum number of severe thermal transients before they must be retired. Each time the engine goes through a startup transient, the life-cycle count for each of the critical components is incremented. This way, part life is tracked as a function of use to facilitate scheduled maintenance.

While these traditional control and diagnostic techniques are time-tested and reliable, advanced techniques provide the promise to meet the challenging requirements of improved fuel efficiency, increased durability and life, decreased life-cycle costs, and improved operations. Using an onboard engine model to meet the challenging control and diagnostics requirements has emerged as the most viable approach. The continuing increase in computer processing capability has reached the point where the use of model-based algorithms for diagnostics and control of aircraft engines has become practical. Previously the complexity of the thermodynamic cycle model has made their use impractical. Model-based technologies offer the potential for creating intelligent propulsion systems—defined as self-diagnostic, self-prognostic, self-optimizing, mission adaptable and inherently robust—that far exceed current systems in performance, reliability, and safety [10].

The concept of model-based control and diagnostics is shown in Fig. 13. The engine model is driven with the measured engine inputs (shown here as control signals u but usually also includes environmental conditions) to obtain the predicted engine outputs \hat{y} . The predicted outputs are differenced with the measured engine sensor outputs y to form residuals, which can be used to diagnose engine health and adapt the model.

Transition to model-based control can occur in several ways. First, faults can be accommodated by changing the control laws, in a predetermined way, when a fault is detected. The changes are designed to, at a minimum, take the engine to a safe state and preferably allow the engine to operate safely with best, although probably degraded, performance. Second, the model allows the loop to be closed on unmeasured values (e.g., thrust, stall margin, etc.) for which there is no sensor, that is, virtual sensors. Finally, in its most advanced form, the model

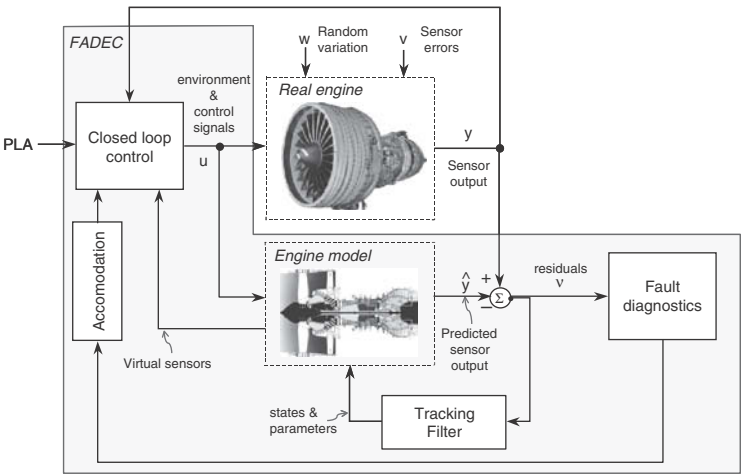


FIG. 13 Model-based control and diagnostics concept.

is used directly in the control enabling the control to automatically adjust as the model adapts to the mission, deterioration, faults, weather, etc. Here the control can be designed to maximize performance without excessive conservatism.

The single-input single-output (SISO) control approach discussed so far is simple and adequate for conventional engines where the main fuel is indeed the dominant actuator and the control requirements are not very stringent. However, it is inadequate for engines where there are multiple actuators with significant interaction between the actuators and the engine outputs to be controlled, for example, a variable-cycle engine or advanced commercial/military engines. There is extensive research that has been done on application of multi-input multi-output (MIMO) control design approaches to engines; see [11] for example. The discussion in this section is applicable regardless of whether the control architecture is SISO or MIMO.

This section provides an overview of the progress and challenges in using a model-based approach to intelligent control and health monitoring of aircraft engines. This section is organized into three subsections: model-based control, onboard condition monitoring, and adaptive control. The focus of these technologies is on developing algorithms that are implemented in the FADEC in the form of software without any hardware changes on the engine in terms of any additional control effectors/actuators.

A. MODEL-BASED CONTROL

One of the major challenges in implementation of model-based control is to have the model reflect the actual condition of the engine. Historically a single, fleet average model is used based on known or expected fleet average performance of a particular engine type. Use of this average model to predict the performance of individual engines within a fleet results in varying model errors, corresponding to how each engine deviates from the average. The resulting model error couples with normal flight variations in the input conditions to cause systematic error in the residuals, which limits the capability of the model to accurately predict the unmeasured variables. Also, if a fault occurs in the engine that is not appropriately reflected in the model, it can cause significant errors in estimation of the unmeasured variables. Thus, it is important to ensure that the model being used for control reflects the true condition of the engine. Usually a “tracking filter,” typically a Kalman filter, is used to estimate model parameters related to deterioration, called health parameters, causing the model to “track” the individual engine over time. Additionally, the tracked parameters can be monitored to provide an indication of the health of the components of the engine, for example, the level of deterioration (see Onboard Condition Monitoring Section that follows). Typically, abrupt faults are detected with the diagnostics algorithm and slowly changing health assessed by monitoring these tracked parameters.

Reference [12] provides an excellent example application of model-based control to aircraft engines. However, as discovered during the early research on

model-based engine control, one of the challenges in engine application is that the number of health parameters to be estimated generally exceeds the number of sensors available for the tracking filter design, which poses an underdetermined estimation problem. A common approach to address this shortcoming is to estimate a subset of the health parameters, referred to as “model tuning parameters.” Although this approach enables the Kalman filter to tune the onboard model so that its outputs track measured (sensed) engine outputs, there can be significant error in estimation of unmeasured engine outputs because the impact of the entire set of health parameters will not be accurately represented within the Kalman filter model.

Recently an innovative methodology [13] has been developed at NASA GRC that creates a tuning parameter vector defined as a linear combination of *all* health parameters and of appropriate dimension to enable Kalman filter estimation. Selection of this tuning parameter vector is performed using a multivariable iterative search routine that minimizes the theoretical mean-squared estimation error in the parameters of interest. The new methodology was validated in simulation using an aircraft turbofan engine model. The simulation results demonstrated that applying the enhanced tuning parameter selection methodology resulted in a significant reduction in average estimation error compared to the conventional approach of selecting a subset of health parameters as tuners. Figure 14 shows a

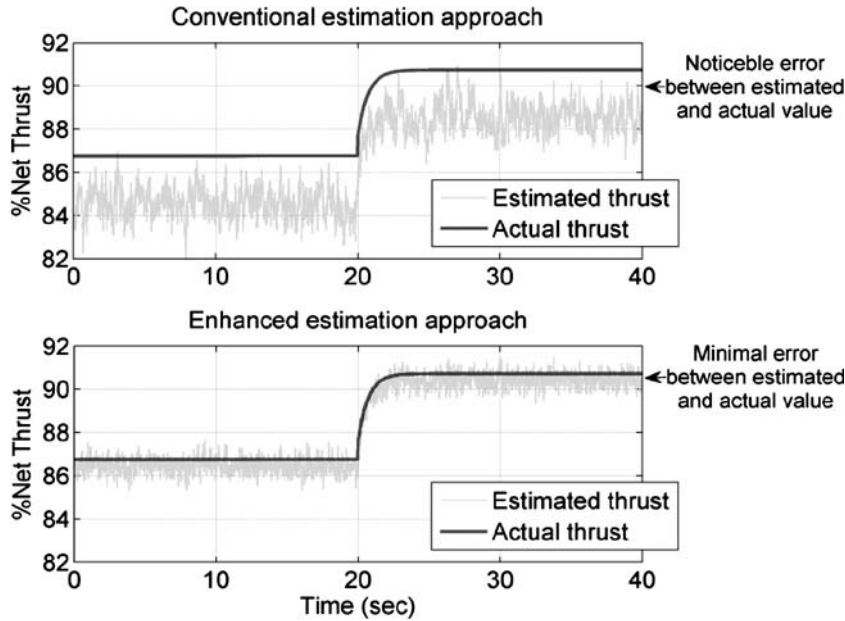


FIG. 14 Thrust estimation accuracy comparison—conventional vs enhanced (optimal) tuning parameter selection.

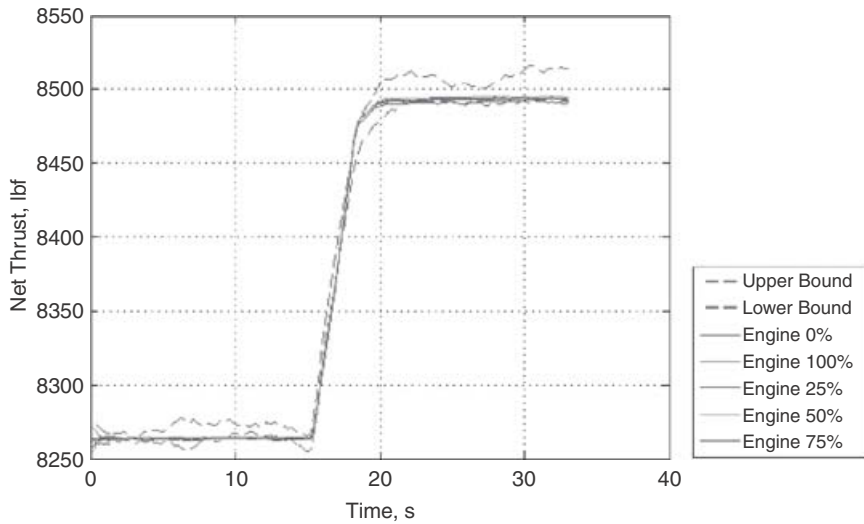


FIG. 15 Thrust response comparison—EPR control vs direct thrust control.

comparison of engine thrust estimates from an engine model using the conventional approach of selecting a subset of health parameters for tuning the onboard model vs the recently developed optimal tuner selection approach. With the optimal tuner selection approach, both the variance and the steady-state error in the thrust estimation are significantly reduced.

With the optimal tuner selection approach developed in [13], sufficient estimation accuracy of the unmeasured variables can be obtained to allow for implementation of direct control of thrust and stall margins using the model estimated value. Research on this application is currently ongoing at NASA GRC. Figure 15 shows a preliminary result from this research. The optimal tuner approach was applied to the simulation of a modern commercial high-bypass turbofan engine. Response of the engine from throttle input to thrust was compared for the baseline case of controlling the engine pressure ratio (EPR) using measured pressures vs closing the loop on the estimated thrust for direct thrust control. In Fig. 15, the dashed red line shows the bounds for thrust response with the EPR control as the engine degrades with usage. With direct thrust control, the thrust response is maintained within a very tight bound regardless of the engine condition.

B. ONBOARD CONDITION MONITORING

With increased performance and affordability of onboard computing capability, there is a recent trend, at least in large civil engines, to have a standalone, engine-mounted engine monitoring unit (EMU). Such an EMU has access to continuous data and can process large volumes of data, which are impractical to transfer to the

ground. The EMU can perform fault diagnostics using real-time data and is typically used to communicate information to the maintenance personnel about any corrective actions that might be needed [14]. The reasons for having EMU installed separately from the FADEC on the engine are as follows: 1) different level of criticality—with the FADEC having highest criticality level and requiring redundancy and flight-critical software level verification and validation, and the EMU being mostly single channel with non-flight-critical software level; and 2) different life cycle—FADEC needs to be mature at engine certification while the EMU can be installed with some initial functionality, and diagnostic algorithms can evolve with experience from the engine operation.

The focus of this section is on onboard condition monitoring from the perspective of gas path performance diagnostics, which is a subset of all of the capabilities embedded in an EMU. The gas path performance diagnostics involves estimating the values of specific variables associated with the gas path components (components shown in Fig. 2) or changes in these values that might indicate a fault. It is important to distinguish between normal changes in the general health of the engine gas path components and sudden faults. The general health of the engine is equivalent to its level of degradation or effective age and is the baseline from which changes are measured. In the gas path, the health condition of each component is defined by its efficiency and other parameters that change slowly with degradation or show abrupt changes due to sudden faults such as foreign object damage. The information flow diagram in Fig. 16 indicates how damage and wear is related to degraded performance [15]. Online estimation of unmeasurable variables is the basis for many fault detection and isolation approaches [16]. However, in general, there are not enough engine sensors available to allow estimation of these health parameters in flight. Many linear and non-linear techniques have been applied to this problem, but without the addition of diagnostic sensors, the problem will remain. As discussed earlier, the estimation problem arises because the number of health parameters exceeds the number of

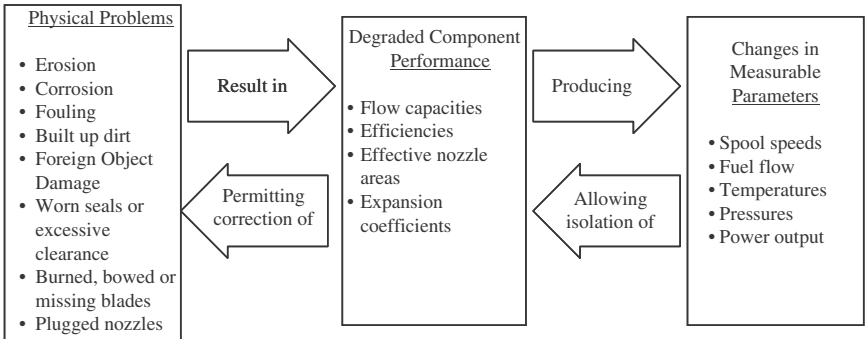


FIG. 16 Diagram indicating information flow for turbine engine gas path diagnostics.

measurements, which means that the problem is underdetermined and thus the health parameter shifts cannot be uniquely determined. The problem is further complicated by estimation errors due to model mismatch, noise, and sensor bias.

As discussed under the model-based control section, the emerging approach to onboard condition monitoring is to use an onboard real-time model to estimate the health and fault status of the engine. When a fault occurs in an engine, a pattern, typically called the signature, occurs in the sensor residuals. The job of the diagnostics algorithm is to detect and identify unique signatures in the residuals (the difference between the sensed value and the model predicted value—see Fig. 13). The presence of random (e.g., sensor noise) and systematic errors limits the sensitivity of fault detection, that is, the size of the fault that can be detected.

The random sensor error cannot be eliminated, but the systematic errors in the average model can be eliminated if an individual model is employed. This is done using the tracking filter approach discussed in the preceding section. Reducing the systematic error in the residuals (measured minus model predicted outputs) effectively increases the signal-to-noise ratio (SNR) of the detection system, where the systematic error is part of the “noise.” The increased SNR results in improved detection capability by allowing either greater fault sensitivity through smaller detection thresholds or lower false alarm rates for existing thresholds.

Fault detection is structured around classical techniques in the signal processing field, for example, multiple model hypothesis testing (MMHT). Other techniques such as fuzzy logic and neural networks can also be employed, alone or in combination, and their results fused [17]. For the MMHT case, one approach to fault detection is to use Bayesian probability computations based on maximum likelihood considerations applied to the residuals, along with any available a priori reliability information. The residuals represent information from multiple locations throughout the engine, and fusion of the information is accomplished through use of multivariate likelihood functions.

Fault isolation is achieved by extending the above fault detection technique to a whole set of specific engine fault candidates through a multiple model hypothesis test structure [18]. Figure 17 illustrates the Bayesian probability-based MMHT algorithm with a one-dimensional (i.e., one sensor) example, where L_i is the likelihood of the residual Δ , given fault i , S_i is the signature of fault i , $prior_i$ is the a priori probability of fault i occurring, and $prob_i$ is the probability of fault i given the observed residual Δ .

Generally, this requires a model of the unfaulted engine, and one for *each* candidate engine fault to generate residuals for each candidate hypothesis (no fault and each fault). The ensemble mean of the residuals for each fault model, when run on an unfaulted engine, defines the physics-based signature of each particular fault. Under steady-state conditions, the individual fault signatures are constant offsets from the no fault signature, leading to a simplified implementation running just the unfaulted engine model. The residuals for the faulted models are formed by adding the precomputed signatures to the unfaulted residuals. Similar techniques exist to simplify the implementation when the assumption

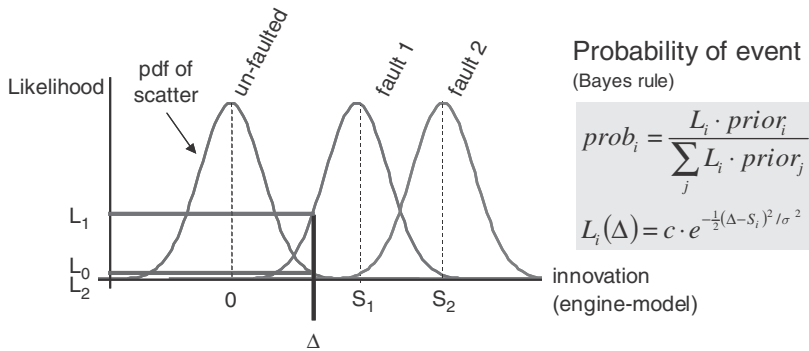


FIG. 17 Multiple model hypothesis testing: one-dimensional example.

of steady state is not valid, although in practice the steady-state signatures are often applied transiently.

For engine gas path diagnostics, the objective is to detect faults as early as possible. To achieve this objective, the online algorithm continuously monitors engine outputs for anomalous signatures induced by faults. The online algorithm must address a challenge in achieving reliable performance. This challenge arises from the fact that the measured engine outputs are influenced not only by faults but also by engine health degradation. Engine health degradation is a normal aging process that all aircraft engines will experience, and therefore it is not considered as a fault. Without a capability to discern the difference between fault-induced and degradation-induced measurement shifts, the online algorithm eventually loses its diagnostic effectiveness as the engine degrades over time. An approach that addresses this challenge consists of integrating the online algorithm with an off-line trend monitoring algorithm [19]. The objective of the off-line algorithm is to trend engine health degradation over the engine's lifetime. The off-line algorithm periodically estimates engine health degradation based on steady-state engine output data recorded during flight. The estimated health degradation is used to update the health baseline (design health condition) of the online algorithm. Through this update, the online algorithm becomes aware of health degradation, and its effectiveness to detect faults can be maintained while the engine continues to degrade over time. This approach is shown in Fig. 18 where OBEM stands for onboard engine model.

One method of implementing the online fault detection algorithm is based on the hybrid Kalman filter (HKF) approach. The HKF is a hybrid of a nonlinear onboard engine model (OBEM) and the linear Kalman filter equation. The main advantage of the HKF over the conventional piecewise linear Kalman filter (PLKF) is that the health baseline is updated through a relatively simple procedure by feeding the estimated health degradation values into the OBEM while the PLKF may require a complete redesign for each new health baseline.

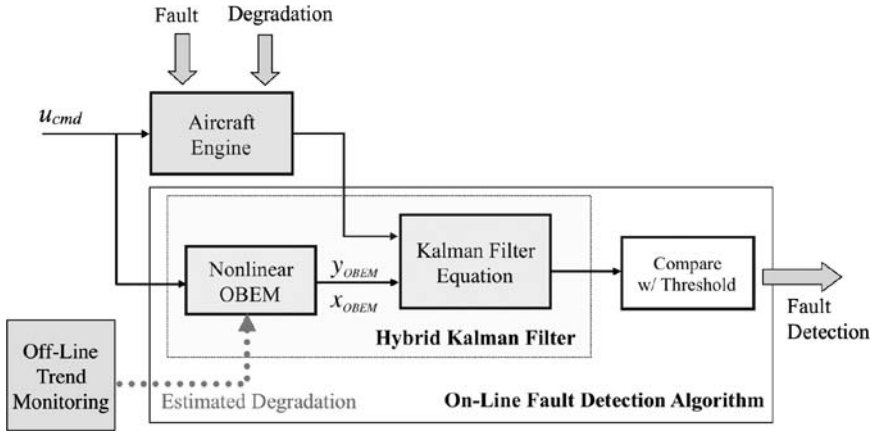


FIG. 18 Integrated off-line and online engine gas path diagnostics.

The diagnostics algorithm was evaluated using an engine simulation representative of a modern commercial turbofan engine. Evaluation results presented in [19] showed that the approach of periodically updating the onboard model resulted in significantly reduced false alarms, as compared to the case of using the baseline onboard model for fault detection as the engine degraded, and also maintained the fault detection performance throughout the engine operating life.

C. ADAPTIVE CONTROL

The traditional engine control logic consists of a fixed set of control gains developed using an average model of the engine. Having an onboard engine model that “adapts” to the condition of the engine opens up many new possibilities. The control logic can adapt to maintain desired performance in the presence of engine degradation or to accommodate faults in a way such as to maintain optimal performance or tradeoff performance for remaining useful on-wing life. An emerging technique for such an adaptive engine control is the model predictive control (MPC) approach. MPC solves a constrained optimization problem online to obtain the “best” control action, based on a tracked engine model, constraints, and the desired optimization objective. The ability to account for the constraints explicitly in the controller design is a key benefit of MPC in contrast with other control algorithms and allows addressing key engine operability and safety constraints directly, for example, speed and temperature limits and stall margin limits. Also, these constraints can be easily modified for increased/decreased conservatism, based on the operation mode or fault condition. Moreover, because MPC solves an optimization problem online, the optimization objective can be modified based on the operation mode, for example, minimize fuel consumption during cruise, minimize turbine temperature for increased life, minimize emissions, etc.

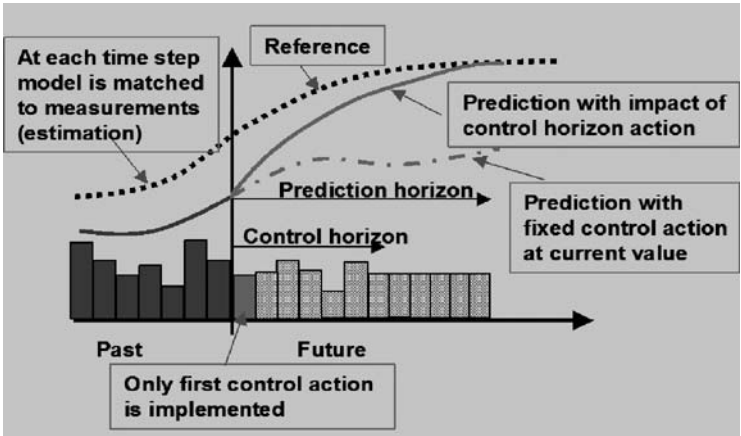


FIG. 19 Model predictive control implementation.

Figure 19 shows the overall approach for MPC. At each time sample, the non-linear engine model is linearized about the current operating point, and the resulting linear state-space model is used to formulate and solve a finite-horizon constrained optimization problem. In this way, the optimal control profile u_k, \dots, u_{k+n} is obtained while enforcing all input and output constraints over the horizon. However, only the first sample of this optimal control profile (i.e., u_k) is implemented, and the whole process is repeated at the next time sample with correspondingly shifted control/prediction horizons.

Figure 20 demonstrates the kind of performance improvements that a MPC-based MIMO controller can provide over a traditional SISO engine control. The figure shows time responses for an engine controlled by a SISO controller

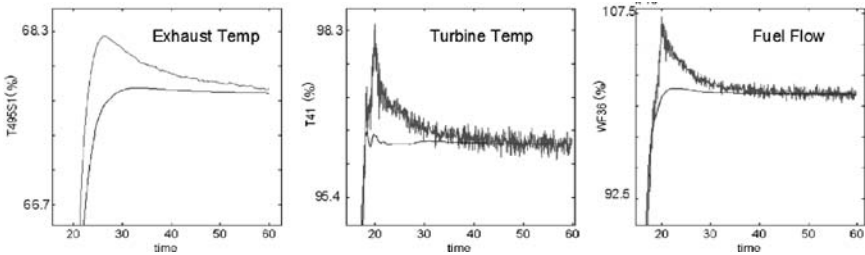


FIG. 20 Improvements of MPC-MIMO controller over SISO controller for idle-takeoff transient. Reduction in peak temperatures extends engine life. SISO actuator is fuel flow, MIMO actuators are fuel flow, VSV, and bleed-valve. The figure shows FADEC/SISO vs MPC/MIMO.

(the current FADEC) and by a MPC during an idle to takeoff transient. The figure shows some key variables—exhaust temperature, turbine temperature, fuel flow—illustrating substantial peak reductions in all of them, which translate into extended engine life. The improvement comes mainly from the fact that the current FADEC is in essence an SISO controller, which drives fuel flow ignoring the interaction between all actuators. The MPC, on the other hand, is multivariable and thus accounts for the interaction between multiple engine actuators and avoids the overshoot in fuel flow. The actuators used by the MPC are fuel flow, VSV, and bleed valve. More details on applications of MPC to jet engines are available in [20]. Note that although performance benefits of MPC-based engine control have been demonstrated in simulations, there are various challenges to be overcome for implementation of MPC-based control logic on actual production engines. One of the major issues is the extensive computations required for MPC implementation, which makes it challenging to implement such algorithms for real-time control computation on onboard computers.

VI. CONCLUSION

Intelligent control and health management technology for aircraft propulsion systems is much more developed in the laboratory than in practice. With a renewed emphasis on reducing engine life-cycle costs, improving fuel efficiency, increasing durability and life, etc., driven by various government programs, there is a strong push to move these technologies out of the laboratory and onto the engine. This chapter described some specific technologies under development that will enable an aircraft propulsion system to operate in an intelligent way—defined as self-diagnostic, self-prognostic, self-optimizing, and mission adaptable. These technologies offer the potential for creating extremely safe, highly reliable systems. The technologies will help to enable a level of performance that far exceeds that of today's propulsion systems in terms of reduction of harmful emissions, maximization of fuel efficiency, and maximization of engine on wing life while improving system affordability and safety. The chapter provided a brief description of state-of-the-art engine control (system level) and inherent limitations in current architecture. Subsequently, objectives for intelligent engine control were described including adapting performance to engine degradation, extending on-wing life, and improving fault detection, isolation and accommodation (FDIA), prognostics, and accommodation via corrective action. An overview of approaches to achieve these objectives was provided. The focus of advanced control schemes is on developing algorithms that are implemented in the full authority digital engine control (FADEC) in the form of software without any hardware changes on the engine in terms of any additional control effectors/actuators. Although the current engines continue to be operated with simple proportional plus integral single-input single-output control loops, the research summarized in this chapter is paving the way for infusing more

intelligence into engine control and diagnostics through the concept of using an onboard engine model to assess the condition of the engine, detect and isolate gas path faults early to prevent the situation from getting safety critical, and improve fuel efficiency through direct control of parameters of interest.

REFERENCES

- [1] Jaw, L. C., and Garg, S., "Propulsion Control Technology Development in the U.S. – A Historical Perspective," International Symposium on Air-Breathing Engines, ISABE-2003-1073, Sept. 2003.
- [2] Spang, H. A., III, and Brown, H., "Control of Jet Engines," *Control Engineering Practice*, Vol. 7, 1999, pp. 1043–1059.
- [3] DeCastro, J. A., Litt, J. S., and Frederick, D. K., "A Modular Aero-Propulsion System Simulation of a Large Commercial Aircraft Engine," AIAA Paper 2008-4579, July 2008.
- [4] Jaw, L. C., and Mattingly, J. D., "Aircraft Engine Controls: Design, System Analysis and Health Monitoring," AIAA Education Series, AIAA, Reston, VA, 2009.
- [5] Urban, L. A., "Gas Path Analysis Applied to Turbine Engine Condition Monitoring," AIAA Paper 1972-1082, Nov.-Dec. 1972.
- [6] Mattern, D. L., Jaw, L. C., Guo, T.-H., Graham, R., and McCoy, W., "Using Neural Networks for Sensor Validation," AIAA Paper 98-3547, July 1998.
- [7] Guo, T.-H., Chen, P., and Jaw, L., "Intelligent Life-Extending Controls for Aircraft Engines," AIAA Paper 2004-6468, Sept. 2004.
- [8] Litt, J. S., Sowers, T. S., and Garg, S., "A Retro-Fit Control Architecture to Maintain Engine Performance with Usage," AIAA Paper 2007-1338, Sept. 2007.
- [9] Culley, D., Garg, S., Hiller, S. J., Horn, W., Kumar, A., Mathews, H. K., Moustapha, H., Pfoertner, H., Rosenfeld, T., Rybarik, P., Schadow, K., Stiharu, I., Viassolo, D. E., and Webster, J., "More Intelligent Gas Turbine Engines," RTO-TR-AVT-128, Research and Technology Organisation, North Atlantic Treaty Organisation, BP 25, F-92201 Neuilly sur-Seine Cedex, France, April 2009, <http://www.rta.nato.int/Pubs/RDP.asp?RDP=RTO-TR-AVT-128>.
- [10] Litt, J. S., Simon, D. L., Garg, S., Guo, T.-H., Mercer, C., Millar, R., Behbahani, A., Bajwa, A., and Jensen, D. T., "A Survey of Intelligent Control and Health Management Technologies for Aircraft Propulsion Systems," *Journal of Aerospace Computing, Information and Communication*, Vol. 1, Dec. 2004, pp. 543–563.
- [11] Frederick, D. K., Garg, S., and Adibhatla, S., "Turbofan Engine Control Design Using Robust Multivariable Control Technologies," *IEEE Transactions on Control Systems Technology*, Vol. 8, No. 6, Nov. 2000, pp. 961–970.
- [12] Adibhatla, S., and Lewis, T., "Model-Based Intelligent Digital Engine Control (MoBIDEC)," AIAA Paper 97-3192, July 1997.
- [13] Simon, D. L., and Garg, S., "Optimal Tuner Selection for Kalman Filter-Based Aircraft Engine Performance Estimation," *Journal of Engineering for Gas Turbines and Power*, Vol. 132, No. 3, March 2010.
- [14] Tanner, G. F., and Crawford, J. A., "An Integrated Engine Health Monitoring System for Gas Turbine Aero-Engines," *IEE Colloquium*, Vol. 3, 2003, pp. 31–42.

- [15] Urban, L.A. "Parameter Selection for Multiple Fault Diagnostics of Gas Turbine Engines," *Journal of Engineering for Power*, Vol. 97, No. 2, April 1975, pp. 225–230.
- [16] Li, Y. G., "Performance-Analysis-Based Gas Turbine Diagnostics: A Review," *Journal of Power and Energy*, Vol. 216, No. 5, 2002, pp. 363–377.
- [17] Rausch, R., et al., "Towards In-Flight Detection and Accommodation of Faults in Aircraft Engines," AIAA Paper 2004-6463, Sept. 2004.
- [18] Kobayashi, T., and Simon, D. L., "Application of a Bank of Kalman Filters for in-Flight Aircraft Engine Sensor Fault Diagnostics," *Proceedings of the ASME Turbo Expo*, 2003; also Paper # GT-2003-38550, June 2003.
- [19] Kobayashi, T., and Simon, D. L., "Integration of On-Line and Off-Line Diagnostic Algorithms for Aircraft Engine Health Management," *Journal of Engineering for Gas Turbine and Power*, Vol. 129, Oct. 2007, pp. 986–993.
- [20] Brunell, B. J., Viassolo, D. E., and Prasanth, R., "Model-Based Control Adaptation and Optimization for Critical Mission Requirements," *Proceedings of the ASME Turbo Expo*, 2004; also Paper GT-2004-53780, June 2004.

Genetic Fuzzy Controller for a Gas-Turbine Fuel System

Andrew Vick* and Kelly Cohen†

University of Cincinnati, Cincinnati, Ohio 45221

I. INTRODUCTION TO GENETIC FUZZY SYSTEMS

With the dawn and advancement of the industrial age, means for controlling mechanical systems and processes have been sought. From early electronic circuits to modern digital controls, systems have been developed to provide automation, improve accuracy, increase efficiency, or expand capability. This trend is expected to continue, as new technological challenges are undertaken.

Control systems cover a broad spectrum of applications, as well a broad spectrum of involvement in a given system. Some control systems function merely in watchdog function roles, monitoring critical parameters in a system and taking mitigating action in the event an unsatisfactory condition is detected. In other systems, the control system is absolutely critical to the performance and capabilities of the system. In fact, many modern tactical aircraft are designed to be aerodynamically unstable, and when augmented with a well-designed control system provides enhanced flying qualities and increased operational capabilities to give an edge in their designed mission.

In an age driven by technology, where systems are constantly being pushed to meet new challenging requirements, advancements in control system design are needed to keep up. We want our mechanical systems to be faster, more accurate, and more robust. We want our systems to be intelligent, to be able to analyze current conditions and objectives, and to make decisions on appropriate behavior, instead of just applying generic global behavior. To this end, intelligent control techniques are gaining traction and increased focus.

Fuzzy logic control is one such intelligent control technique and will be a pillar of this study. This nonlinear control design technique provides significant benefits in terms of flexibility and optimization space. When coupled with the ability to capture expert or heuristic knowledge, and the ability to tune behavior in local envelopes of the operating space, fuzzy logic can be an indispensable control

*Graduate Student, School of Aerospace Systems, College of Engineering and Applied Science; also, General Electric Aviation, Control Systems Design; currently Lead Control Systems Design Engineer, General Electric.

†Associate Professor, School of Aerospace Systems, College of Engineering and Applied Science.

design tool in many applications. Fuzzy logic control also possesses inherent robustness properties due to having knowledge-based properties, making them good candidates for stochastic systems.

As a result of the flexibility of fuzzy logic controllers, one challenge facing control designers is the tuning process. Fuzzy logic controllers can have a variety of handles to impact performance, from the fuzzy input and output sets to the governing rule base. Genetic algorithms, a branch of evolutionary algorithms, will be utilized in this study to provide an autonomous guided search of the design space to develop a more optimized solution against the design requirements. This approach uses a search scheme based on evolutionary principles, a “survival of the fittest” type technique, including genetic recombination and variation.

In addition to ensuring a controller design meets performance requirements and objectives, a designer will also want to make sure the controller will be robust enough to meet those requirements in all operating conditions. A controller may perform admirably for a given mechanical system design, but will it uphold that performance when part-to-part variation due to manufacturing capabilities is taken into account? Will it stand up to all operating conditions and system degradation? A study in controller robustness is needed to answer these questions, and in this study a stochastic robustness analysis will be performed to ensure the controller maintains adequate performance in an uncertain environment.

A. FUZZY LOGIC CONTROL

Fuzzy sets were introduced by Zadeh [1] as a means of handling imprecision in a classification process. Membership to a fuzzy set is defined by membership functions, continuous functions with values between 0 (no membership) and 1 (full membership). Zadeh suggested these sets for use in problems where precision might not be possible or necessary, and the desired outcome could still be attained.

In addition to the definitions associated with fuzzy sets and membership functions, Zadeh also extended the definitions of operations associated with set theory into fuzzy set theory, such as union and intersection. The union of two fuzzy sets was defined as the max of the two sets, while intersection was defined as the min. With these definitions, the behavior for crisp or singleton sets is preserved, yielding the same result as union or intersection in the classical sense.

Membership functions are a mathematical representation for the degree to which something belongs to a fuzzy set. Say you wanted to classify a given temperature ($[0^\circ\ 100^\circ]$ °C) using the categories *cold*, *warm*, and *hot*. If using crisp sets, this would be difficult to do. Where would you draw the line between a warm temperature and a hot temperature? The difference between any adjacent categories would only be 1°C , barely discernible to most human sense observations, yet the temperatures would be placed in two distinct categories. Fuzzy logic can help deal with this type scenario, by defining some membership to both adjacent categories.

Using the example just discussed, Fig. 1 illustrates the membership functions for the three different temperature categories. To illustrate how membership

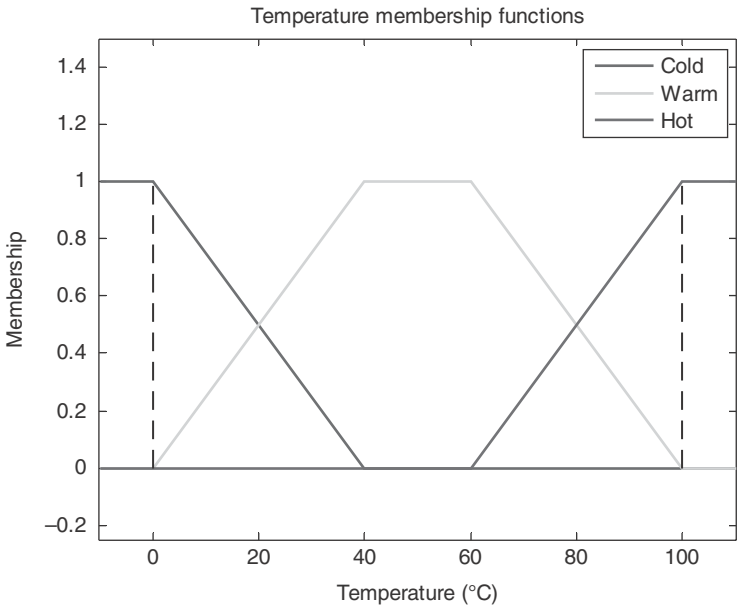


FIG. 1 Sample membership functions for temperature.

functions operate, the membership towards the different categories will be determined for a single temperature, 70°C. Moving to this temperature on the horizontal axis, the corresponding membership to each category can be found by determining the membership function value at the temperature of interest. As illustrated in Fig. 2, this temperature has 0 membership in the category of *cold*, belongs to *warm* to degree 0.75, and shows a 0.25 membership to the category of *hot*. This shows the temperature of interest has some membership to both the *warm* and *hot* categories, but still shows the temperature is more associated with a *warm* temperature. The membership function values can be used in some sort of weighting scheme for determining the output of a system based on this input temperature, or also can be used in a fuzzy logic control system, which will be demonstrated shortly.

Membership function categories and their values are up to the designer and could be adjusted based on system needs. In the preceding example, trapezoidal membership functions were used (truncated for limits), but membership functions can take a number of different forms. Isosceles triangles are also fairly common, but it is possible to use nonisosceles triangles as well. Gaussian distributions and exponential distributions are also available for use to the designer following scaling, as well as any custom functions with values over the interval from [0 1]. Mamdani expanded on Zadeh’s use of fuzzy sets to develop a control system based on fuzzy logic principles [2]. A fuzzy logic control system can operate in a

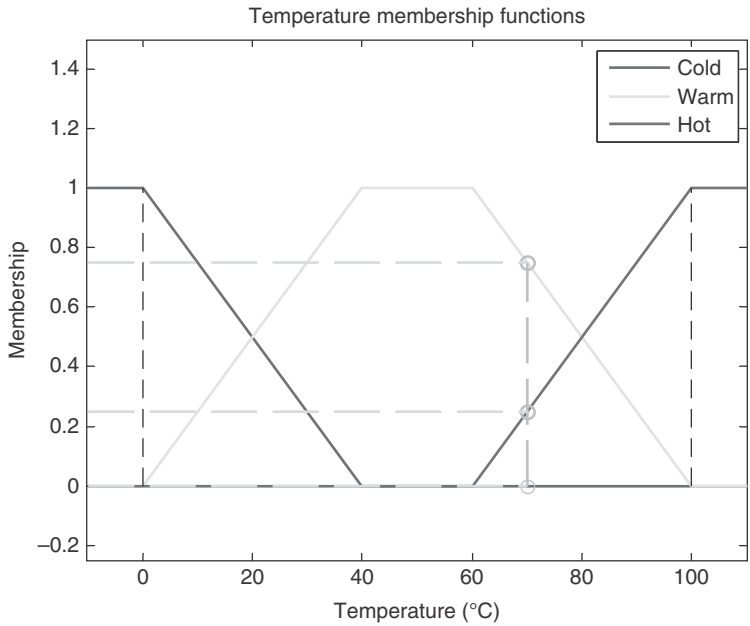


FIG. 2 Membership function evaluation for set temperature.

closed-loop environment similar to other control techniques. The system takes inputs from the system, makes decisions on the desired response based on classification of those inputs, and determines appropriate outputs. The result is a fairly simple nonlinear control system that can be tuned for desired performance.

The basic structure of a fuzzy logic controller is illustrated in Fig. 3. This figure illustrates the four main components of this type of controller: the fuzzifier, rule base, defuzzifier, and the inference engine. These components work in tandem to provide closed-loop control for a given system of interest. Details on each of these components and their roles are discussed next.

Fuzzifier: The purpose of the fuzzifier is to transform the crisp inputs into the fuzzy logic controller into fuzzy inputs. The inputs to this type of system can range from sensed quantities, estimated quantities, or even system demands for a desired state. The fuzzifier uses a series of membership functions to categorize the crisp inputs into their respective fuzzy set representations. The fuzzifier determines which fuzzy sets are “active” (nonzero membership) based on each of the crisp inputs and also to which degree. The membership function example using the temperature scale could be viewed as a sample fuzzifier. The crisp input coming in would be the temperature, measured using a sensor or estimated from a mathematical model. As shown, the output would be the membership function values associated with each fuzzy set.

Fuzzifiers must provide *coverage* of the input space; every potential input to the fuzzy system must result in membership to one or more of the possible fuzzy sets. This helps ensure *completeness* for the fuzzy logic control system, to be discussed in more detail in the section on rule bases. *Computation* is also an important consideration when designing a fuzzifier. For each fuzzy set that becomes active based on a given input, the mathematics of the inference engine becomes more involved as a result, especially when considering controllers with multiple inputs, with different fuzzy sets associated with each. Therefore, it is standard practice to ensure that only two fuzzy sets are active at a time for each input, leading to the guideline that only adjacent fuzzy sets should be allowed to overlap in their membership function distributions.

Rule Base: The rule base of a fuzzy logic controller can be thought of as the “brains” of the control system, or its means of determining what actions to take based on a given set of inputs. A rule base is made up of series of IF-THEN rules corresponding to the fuzzy inputs and leading to the fuzzy outputs. The rules can be developed using knowledge from experts or operators in the field, as well as historical experience. This allows for control system behavior to be developed without extensive expertise in control systems, or it can also serve as a means for adapting a controller based on system behavior. Cordon and others suggest methods for tuning rule bases to achieve desired performance, but this approach is beyond the scope of this study [3].

As a simple example, a sample rule base will be developed for an automobile’s cruise control system. After all, most drivers have a basic understanding of a cruise control system and would be able to develop basic rules for governing behavior. This example will use the difference (delta) between the desired set speed and the current speed as the input, classified into fuzzy sets *low*, *even*, and *high* by a fuzzifier. Using this information, a sample rule could be “IF speed delta is *low*, THEN *accelerate*.” Here, the term *accelerate* corresponds to a fuzzy output set, which would be defined in the defuzzifier. With little thought, most drivers would be able to develop the set of rules illustrated in Table 1.

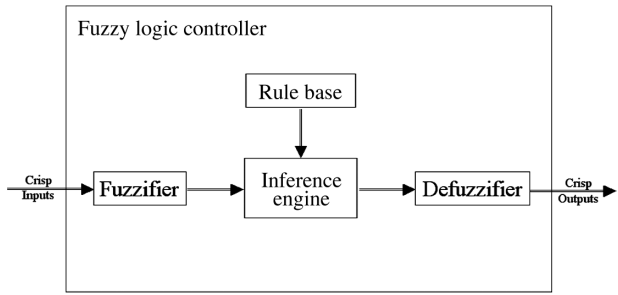


FIG. 3 Fuzzy logic controller structure.

TABLE 1 SIMPLE AUTOMOBILE CRUISE CONTROL RULE BASE

Input	Output	Rule
Low	Accelerate	IF speed delta is low, THEN accelerate
Even	Maintain	IF speed delta is even, THEN maintain
High	Decelerate	IF speed delta is high, THEN decelerate

This rule base is fairly crude and may not deliver adequate performance in all driving conditions. Enhancements to this rule base may involve adding more fuzzy sets to the input and output systems, increasing granularity. For example, modified categories like *very low* or *slightly high* could be added as additional fuzzy sets to the fuzzifier, and the output could be expanded to include additional classes for *accelerate* and *decelerate*, perhaps to provide varying degrees of these actions. Another option would be to include an additional input to the fuzzifier, perhaps a rate term in addition to the speed difference. Examples of rules using these expanded fuzzy sets could include statements such as “IF speed delta is *slightly low* AND speed is *increasing*, THEN *maintain*” or “IF speed delta is *slightly low* AND speed is *decreasing*, THEN *accelerate*.” This could provide a means of differentiating different driving cases, leading to different controller behavior dependent on a given scenario. Although this example is fairly basic and not complete, it should provide insight into the rule-making process and how working knowledge of a system could be captured without the need for expertise in control system design.

Defuzzifier: The defuzzifier fulfills the inverse role of the fuzzifier; it takes the information from the fuzzy output sets and transforms them into crisp outputs that can be used by the control system. This output could be a number of different things, from controller loop gains, to modeled parameters, to the system demands themselves. The typical defuzzifier consists of membership functions similar to a fuzzifier, but the system works in the opposite direction. Instead of working from the input on the horizontal axis to determine the fuzzy set memberships, the active fuzzy sets are used to find a crisp output on the horizontal axis. The active fuzzy sets in the defuzzifier are determined by the rule consequents in the inference engine, and some mathematical approach is used to combine these sets into a single output value. A *centroid* method is one of the more common and simplistic approaches and will be employed in this study and demonstrated next. Numerous other methods have been proposed as alternatives to try to accommodate some of the potential pitfalls or special cases that can arise [4].

Inference Engine: The inference engine is the component that ties all of the other components of the fuzzy logic controller into a complete system. The engine takes the fuzzy inputs from the fuzzifier, checks them against the rules in the rule base, and determines which fuzzy output sets are activated based on exercising the rules. The degree of membership for each of the fuzzy input sets is used to determine the “degree of activation” of the corresponding output sets. Using the temperature

TABLE 2 SIMPLE WATER HEATER CONTROL RULE BASE

Input	Output	Rule
Cold	High heat	IF input temperature is low, THEN apply high heat
Warm	Low heat	IF input temperature is warm, THEN apply low heat
Hot	Maintain	IF input temperature is hot, THEN maintain

classification example just illustrated again, a temperature of 70°C had membership of 0.75 to fuzzy set *warm* and 0.25 to *hot*. Therefore, the output fuzzy set(s) tied to the input set *warm* through the rule base would be activated to degree of 0.75, and those outputs tied to the set *hot* would be activated to degree 0.25. To better illustrate this, consider the temperature classification example as the fuzzifier for a water heater fuzzy logic controller application. The goal of the heater is to maintain temperatures above 70°C, temperatures predominant in the *hot* category. The output fuzzy sets associated with this application may be actions like *maintain*, *low heat*, and *high heat*, activated through the simple rule base depicted in Table 2.

Figure 4 illustrates how the output sets are activated based on the membership values on the input sets. These output sets are normalized on an interval of [0 1], but could be scaled to any appropriate level. The output sets are triggered into action to

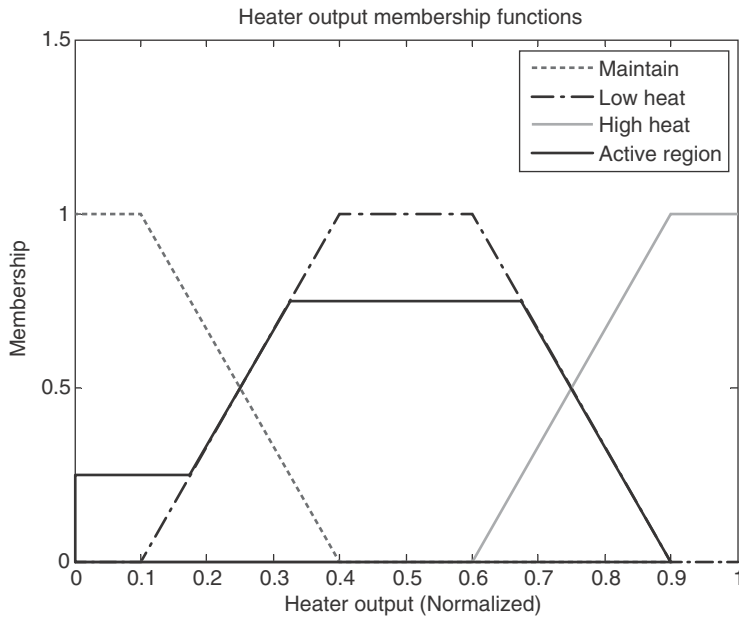


FIG. 4 Output fuzzy sets for water heater example.

the same degree of the membership of the inputs. The resulting area on this figure would represent the combination of the outputs that are active for this set of inputs, and the crisp output would be determined based on this information.

This example used a simple case with single-antecedent rules. It is quite common for rule antecedents to be combined, such as in forms "IF x_1 is A AND x_2 is B, THEN y_1 is C" or "IF x_1 is A OR x_2 is B, THEN y_2 is D." In these cases, the membership function values of the inputs are combined using fuzzy definitions of these operations. The AND operator (intersection) would result in the minimum of the two input membership values, or the product of the two if this alternative definition for fuzzy intersection is used. The OR operator (union) would result in maximum of the two input membership values being used, or any one of a number of alternative definitions [4]. Of course, these operators could be expanded to whatever number of rule antecedents are used for a given rule base system.

As with any new design approach, it will have benefits and drawbacks compared to other design methods. Fuzzy logic controllers allow for a simple implementation of a nonlinear control system design. This approach is also a candidate for embedded systems because of its relative ease for computation. The structure of this type of controller also provides for compartmentalization, leading to the possibility of the controller to be tuned for specific behavior in smaller, local operating regions, instead of having to tune or check the performance over the entire operating envelope following controller adjustments [5]. Fuzzy logic controllers are also able to perform admirably in the face of uncertainty in a given control application, making them good candidates for systems that need to meet higher performance objectives in environments affected by uncertainties.

Fuzzy logic controllers have been explored and implemented for a number of different applications. Cohen et al. develop a controller for a linear second-order system proposed as a benchmark problem by the American Control Conference [6]. This problem consisted of a system with plant uncertainty and had a non-collocated sensor and actuator pair, leading to a nontrivial design problem. They cite a fuzzy logic controller as a good candidate for this application due to its ease of implementation and its inherent robustness characteristics. Also mentioned is the ability of the controller to emulate the minimum-time solution derived from optimal control theory using heuristics in its rule base. The controller in this paper was developed with a Luenberger observer in the loop; however, the current study will have access to necessary control states. Following controller development, the paper also examined the robustness characteristics of the control system through *stochastic robustness analysis*, examined further in Sec. I.C of this chapter.

Nelson and Lakany have studied use of fuzzy logic in the fuel control system for industrial gas turbines [7]. In their paper, they discuss how traditional control system design methods may encounter difficulties in systems that can degrade over time, especially in areas of emissions control. They suggest that fuzzy logic controllers might be a good candidate for this type of environment, lending to their nonlinear properties and robustness in cases where the plant may not be well understood (as in cases of degradation). They develop a fuzzy logic controller

to compare performance against an existing proportional-integral (PI) controller. Through various operational scenarios, performance of the two controllers was comparable, although the PI controller typically had better response and settling times. In some scenarios, however, the fuzzy logic controller did show better performance in terms of exhaust temperature overshoot, which is a critical performance measure in industrial gas turbine operations.

A key comment Nelson and Lakany mention in their paper is the difficulty to tune the fuzzy logic controller to a more optimum design point. They implemented a controller with two inputs, error and error rate (E and \dot{E}), using five fuzzy sets for each. The output, an incremental change to fuel demand, also had five sets, although singleton sets were used. The system used 25 rules in the base for the inference engine. The result is a large number of parameters that can be tuned to affect performance, although some simplifications were made.

By assuming a consistent shape for the membership functions, the five fuzzy sets for each input can be defined by two parameters or, assuming symmetry about the zero axis, can be reduced to one. However, fuzzy sets with such a simple structure may be constraining in terms of optimization. The authors also proposed an alternative fuzzy set definition, using symmetry about the zero point in the input space, but using nonisosceles triangular membership functions where each fuzzy set is allowed to be a different size. This yields a more flexible fuzzy set definition that can be better tuned for performance than a rigid structure with identically shaped membership functions, but still only needs two defining parameters. As illustrated in Fig. 5, x_1 and x_2 can entirely define the membership functions for the five fuzzy sets.

Of course, this approach can be expanded for inputs with more fuzzy sets, adding an additional tuning parameter for each pair of additional sets. The

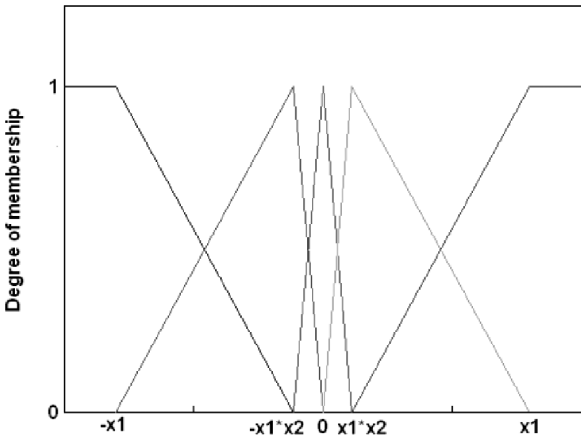


FIG. 5 Nonisosceles fuzzy membership functions [7].

symmetry constraint is common in fuzzy logic controller designs, as it is often desired to have consistent system response when moving in different directions. This approach may still provide some constraints on a truly optimal solution, but it does add flexibility to the design space and ensures that total membership across fuzzy sets always sums to one, which can be an important factor for controller consistency.

A fuzzy logic controller for a fuel system test bench is developed in an article by Zilouchian and others [8]. This test bench is designed to allow for proper simulation testing of a jet-engine fuel delivery system, by controlling the combustor pressure that would be experienced in a given operating condition. The fuel system test bench is currently controlled with a traditional PI controller, but performance shortfalls are noted. This paper looks to fuzzy logic as a solution due to its nonlinear characteristics and the ability for tuning in small operating envelopes (compartmentalization). The authors also make notes about the difficulties in tuning fuzzy logic controllers due to the number of design options. In this paper, they chose to use a consistent structure of seven triangular membership functions with equal size normalized over the range $[-1, 1]$ for both the inputs (error and change in error, E and ΔE) and output (valve positions) and chose to optimize using a scaling parameter up and downstream of the fuzzy sets. This restricts flexibility, but provides a simple structure for a first cut at controller design. The authors went on to develop separate fuzzy logic controllers for “coarse” and “fine” control, utilizing the compartmentalization capabilities of fuzzy logic schemes.

As just discussed, one of the major drawbacks to a fuzzy logic controller cited in various sources is the tuning process. The membership function types and the distributions themselves can present a great number of design options for the input/output fuzzy systems, which can be paralyzing for a designer trying to optimize a given control system. The rule base provides another set of inputs that can be used to alter the performance of the loops. Use of genetic algorithms can help in this area, performing a guided search towards an optimal design and taking some of the burden off of the control system designer. Details of genetic algorithms will be discussed in the next section.

B. GENETIC ALGORITHMS

Genetic algorithms are a class of optimization approaches based on evolutionary principles. Introduced by John Holland in 1975, this meta-heuristic approach is capable of providing an optimal solution based on the premise of “survival of the fittest” [9]. The approach emulates Charles Darwin’s research on natural selection that those members of a population best suited for their environment are more likely to survive and pass on their genetic traits. In a genetic algorithm, members of a population are represented by numeric strings (dubbed *chromosomes*) and evaluated for fitness for a certain function. Similar to natural selection, those that receive higher fitness ratings are more likely to see their chromosomes passed on to subsequent generations of the population, and through an iterative

process, this algorithm will converge to a better performing design. This algorithm also incorporates the ideas of *genetic variation*, which will be discussed further in this chapter.

One of the first steps in developing a genetic algorithm is to determine how members of the population will be represented. Binary strings are the most common and often simplest representation, but it is also possible to represent members of the population with nonbinary, or real-value coded strings. Fixed-length strings are more common, but Cordon and others discuss details of variable-length strings and their benefits and application [3]. For this study, fixed-length binary-coded strings will be employed in the optimization algorithm for their inherent simplicity.

One problem that can arise when using a binary-coded string in the traditional base-two sense is related to the large hamming distances that are realized for successive integers. Integer values 7 and 8 for example have binary representations of 0111 and 1000, respectively, illustrating a difference in every character in the string for a relatively small difference in numeric value. The iterative genetic algorithm process may have a difficult time converging to a true optimized point using basic genetic variations (discussed next) if it would have to cross a large hamming distance en route. Cordon and others propose the use of a *gray code*, in which only hamming distances of one separate successive integers [3]; however, single bit value changes may still have a large impact on the represented numeric value. This study will utilize multiple runs of the genetic algorithm, each with a different random initial population to try to mitigate this shortcoming.

Once the representation format for the members of the population has been determined, the initial population needs to be generated. The number of members in the population is left up to the designer. Larger populations will provide more opportunities to search the design space and provide population variation, but will also take longer to determine performance and the makeup of successive populations. The initial set can be generated in a number of different ways, ranging from purely random generation for each member string, to sparse sets that try to cover the range of the design space, or even initial predetermined candidate sets that correspond to known performance levels for which improvement is desired. Hybridization of these methods can also be utilized to generate an initial population.

With the initial population in place, the next step is to develop a means of assessing the performance of each member for a given function. A common example could be a genetic algorithm that tries to minimize the time for a given process. For this type case, the fitness function could simply be the inverse of the calculated time, yielding higher fitness ratings corresponding to members with the lowest process times. In this study into closed-loop control applications, the fitness function(s) will be based on common controller performance metrics such as response time, settling time, and control efforts.

As future populations are often developed using probabilities of selection based on the fitness ratings, it might be important to ensure that the fitness

values are all positive. Another common pitfall in the optimization process is for a member to stand out from other members in terms of vastly superior fitness rating, which can lead to premature convergence if it occurs in early generations depending on the selection schemes used. Also, in later generations when most members begin to illustrate improved performance, it might make it difficult for any particular member to stand out, slowing the process of convergence to an optimized solution. As a means for mitigating these scenarios, Cordon and others provide suggestions for a number of fitness function scaling methods [3].

With the performance for a current population known, this information can be used to determine the candidate makeup of the next generation. Some selection schemes place direct copies of the top performers of a population into the subsequent population, known as elitist selection, ensuring that best performer of the next generation is always at least as good as the one before it. This approach also ensures that the candidate pool contains the genetic strings of the best known solutions are always available for future generations, immune to being removed due purely to random occurrence. Similarly, discarding the poorest performers is another option, although these candidates may still have some desirable qualities that could become part of future solutions and would tend to be removed from future generations anyway as part of the genetic algorithm process.

Most selection schemes choose candidates from the existing population using a probability measure determined from their fitness. The simplest of these measures is to divide the fitness of a given candidate by the sum of fitness for all candidates, although other probability methodologies have been suggested [3]. Using the simple approach, the candidates can then be thought of as being arranged on a roulette wheel, filling area proportional to their probability level. The wheel is then spun to select a candidate as many times as needed to fill out the full population.

This roulette wheel approach has some inherent flaws, one being the potential for large discrepancy between the expected and actual number of selections for a given candidate. Alternative methods for selection have been proposed, one of which being stochastic universal sampling [3]. In this sampling method, the roulette wheel is only spun once, while pointers spaced equidistantly around the wheel determine the candidates needed to fill the population. There are numerous other schemes that can be employed in the selection process, each trading off simplicity for the likelihood of a better population sample.

With the candidates for the next generation of a population selected from the previous, the genetic variation process begins. Three different processes will be used to introduce genetic variation into the existing candidates: crossovers, mutations, and inversions.

Crossovers involve swapping genetic material between two chromosomes. In this way, optimal solutions are sought in the area of current knowledge in the design space. The simplest of these is the one-point crossover, where a number less than the length of the chromosome strings is chosen at random, and the genetic strings to the right of that number are swapped between two parent



FIG. 6 Illustration of one-point crossover.

chromosomes. This is illustrated in Fig. 6 for two sample binary strings swapping the rightmost two bits.

The two-point crossover is also common and helps alleviate the bias that the latter part of the chromosome strings are more likely to be swapped. In this process, two random numbers are chosen, and the string of characters between the two numbers is swapped in the parent chromosomes, as illustrated in Fig. 7. Of course, this process could be extrapolated for higher-order crossovers, up to the limiting case of the *uniform crossover*, where the child chromosome is developed one element at a time, choosing each character independently from the two parent chromosomes.

Another process to introduce new genetic combinations is mutation. In this process, each individual character in a genetic string is independently subjected to the potential to change value. In binary genetic strings (e.g., Fig. 8), the outcome is simple; 1s turn to 0s and 0s turn to 1s. Whereas crossovers tend to search for new solutions near the current ones, mutations have the possibility to “unlock” new regions of the design space that haven’t been explored. This aids in avoiding premature convergence to local extremes, helping to find the overall optimal solution in the available envelope.

The final means of introducing genetic variation considered in this study is inversion (see Fig. 9). Similar to mutation, this process is another means of exploring the design space. In this process, the binary chromosome string of the least fit candidate(s) is “inverted” such that all 0s become 1s and vice versa. Although this type of operation does not have as strong a tie to genetic processes, it might be a helpful tool in developing new “genetic material.” The premise is that the least fit candidates are far from an optimal solution and that by jumping to a new point in the design space a more optimal solution can be found. This approach may also explore new regions that might not be optimal overall, but might have some

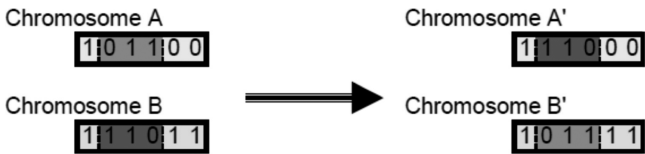


FIG. 7 Illustration of two-point crossover.



FIG. 8 Illustration of binary mutation.

desirable qualities that could become intertwined with existing solutions through the crossover process.

Figure 10 illustrates the entire process of generating a new ten-member population for the next iteration of a genetic algorithm once the fitness of the current population has been determined. This figure is intended for illustrative purposes and is not representative of any data generated for this study. Figure 10a illustrates the chromosomes from the current population, sorted from highest fitness to lowest. Demonstrating an elitist scheme, the top two performers are placed directly into the next population. Also placed into the next population is the binary compliment of the worst performer or an inversion of all of the bits within the chromosome. These three chromosomes will be tested in the next population as is; they are exempt from genetic variation introduced by crossovers and mutations. The base chromosomes for remaining members of the population are selected with some randomness from the current members, typically weighted in some fashion using the current populations' fitness ratings. Figure 10b shows the resulting chromosomes from the selection process.

Following selection, the population (minus elites and inverts) is subjected to crossovers. A random number is generated for each chromosome and compared to the probability number selected for crossovers. For this example, both one- and two-point crossovers were employed. Figure 10c shows the results of the crossover phase. The first column (CO) indicates which type of crossover resulted (one-point, two-point, or none), the second column (PC) indicates the partner chromosome with which genetic material was swapped, and the third (Bits) shows which bits bound the crossover.

Figure 10d illustrates the mutation phase. A random number is generated for each bit of each member of the population and compared to the selected probability for mutations. Each bit that meets the criteria is inverted, as illustrated in the first column of the figure. This phase completes the process for generating the next population, which is illustrated in Fig. 10e.

Genetic algorithms or other types of evolutionary-based techniques have been used as an optimization tool in a number of different applications. Cordon and others illustrate a simple example for path optimization under gravitational



FIG. 9 Illustration of genetic inversion.

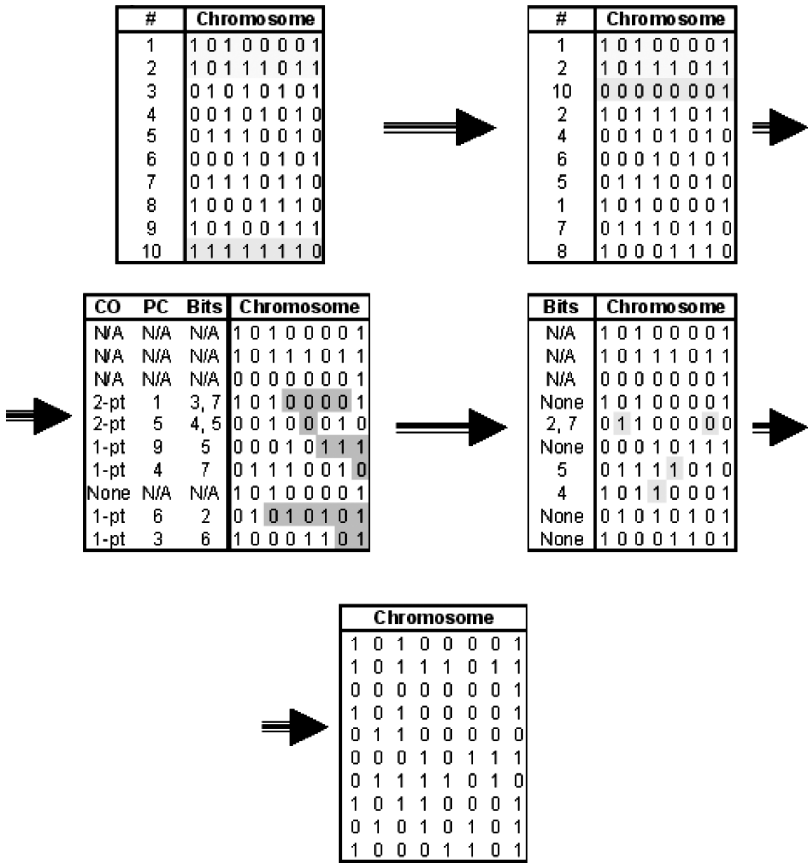


FIG. 10 Illustration of population selection and genetic variation process: a) current population, b) selection, c) crossovers, d) mutations, and e) next population.

influence [3]. The genetic algorithm tries to define the path for minimum time travel between two points. The path is discretized using a number of intermediate points, with each point represented by a six-character binary string using a simple base 2 encoding. Using an initial population of 20 individuals developed randomly, successive populations are determined using an elitist scheme and using only one-point crossovers and mutations as genetic operators. The time to travel between each point is determined and the total summed, using the inverse of this path time as the fitness function. The results were determined from a number of different initial populations, but the minimum paths found were always relatively close to each other, illustrating the robustness of the process.

Nyongesa examines optimization of fuzzy-neural systems using genetic algorithms and other evolutionary strategies [10]. The author describes the benefits of fuzzy logic controllers in areas such as nonlinear behavior and the ability to capture expert or system knowledge. By incorporating neural network capabilities, which are renowned for their knowledge acquisition and adaptive learning capabilities, the paper discusses how fuzzy systems can be further enhanced for robustness and performance in uncertain environments, but points out that optimization of such a system is still cumbersome. Citing genetic and evolutionary algorithms, the author discusses how these techniques can be incorporated to develop fuzzy-neural systems optimized for a given application.

Genetic algorithms have been used to help mitigate the effort associated with fuzzy logic system optimization, collectively known as genetic fuzzy systems. In their paper on diesel engine pressure modeling, Radziszewski and Kekez describe the use of genetic fuzzy systems to develop an accurate model of the complex pressure profiles experienced in the diesel engine cycle [11]. Fuzzy logic was fitting for the application due to its ability to represent nonlinear systems, and also for robustness properties as several different fuel sources were used in the engine modeling. Use of genetic algorithms allows for an easier method for tuning the various fuzzy system parameters (membership functions and rule base) to generate an accurate model of the pressure cycle that was robust enough across several different fuel types. As a fitness function, the authors used a measure of accuracy to experimental data, along with a factor associated with the number of rules in the resulting rule base, such that systems with fewer rules might be rewarded for simplicity. The results consisted of several different genetic fuzzy system models that met desired accuracies over specified ranges.

C. STOCHASTIC ROBUSTNESS ANALYSIS

Controller robustness is a concern in any design application, but this is particularly true in stochastic systems, or systems characterized by some level of uncertainty. The designer will want to ensure the controller performs acceptably throughout the operating envelope, taking into account operating conditions and variation within the system itself. Controller systems designed for a single nominal operating point may have inadequate performance in another operating condition, or may become unstable when hardware part-to-part variation is taken into account. Analysis that considers these factors must be employed as part of the design process to develop a robust system capable of operating in a range of conditions in the face of uncertainty.

Classically, gain and phase margins have been used as a robustness measurement tool. However, in a stochastic system, margins typically deemed adequate can't necessarily guarantee stability. Uncertainty in the plant or measurement system could shift the characteristics of the system, such that a controller that is stable on one version of the system may not be stable on another. Even if stable, key performance characteristics can be lost, leaving a system unable to

meet its requirements. If information about the variation is known or can be estimated, it would be possible to develop a controller to satisfy performance requirements for the “worst-case” system. However, as Stengel and Marrison point out, this analysis can be prohibitively difficult or could lead to an overconservative controller design [12].

The stochastic robustness analysis proposed and demonstrated by Stengel and others [11–18] uses Monte Carlo to simulate a number of different systems. The various sources of uncertainty within the system are represented with distributions, typically uniform or Gaussian, but others are possible if enough information about the variation is known to use a more fitting distribution. For each Monte Carlo run, random numbers are generated using the distribution information to develop a different sample of the system. Controller performance is evaluated for each case, using a straightforward approach that can give a good indication of performance across a fleet or operating range. To evaluate controller performance as part of the stochastic robustness analysis, classical performance measures can be used. Stability is often of greatest concern, but measures like rise time, settling time, or control effort can also be used in conjunction. As Stengel discusses, it is best to mold these performance measures into sets of *pass/fail* criteria, such as *stable/unstable*, or settling times that are *acceptable/unacceptable* [15]. By using this approach, the resulting performance from all of the runs could be congregated using a binomial distribution, and all of the tools associated with this type of distribution can be used in the analysis.

The process for stochastic robustness analysis is illustrated in Fig. 11. Prior to running the robustness analysis, a design for the controller needs to be determined. Oftentimes, design of the controller is an iterative process, trying to balance controller design parameters against performance objectives. Once the controller is in place, the robustness analysis can begin. The first step in the stochastic analysis is to generate a value for each of the identified sources of uncertainty using predetermined distributions (Gaussian, uniform, etc.) based on generated data or knowledge of the system. The collection of uncertain parameters in conjunction with other system information would make up one sample of the system, such as a member of a fleet or a certain operating condition. With the system characteristics fully defined, simulation or mission analysis can be performed as a means of determining performance. Following simulation, stability or other performance measures can be assessed. These steps make up just a single iteration of the analysis, typically referred to as a Monte Carlo analysis. This process is then repeated as many times as the designer would like, with the benefit of increased confidence coming with increased simulations. Once the system and uncertainty distributions are set up, the only cost to additional iterations is computational time, which is becoming less of an issue with increases in processing capabilities.

Once all of the Monte Carlo runs have been completed, the performance of the controller with respect to the various selected measures for each run can be combined to develop an overall sense of robustness across the simulated conditions. If

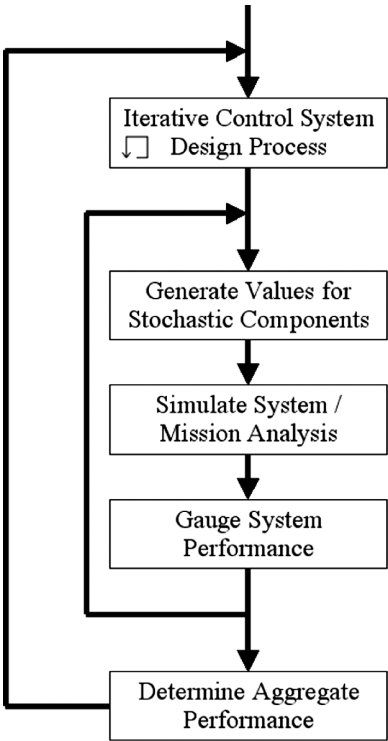
FIG. 11 Illustration of stochastic robustness analysis process.

performance against each measure were assessed in a binary sense (pass/fail), then the binomial distribution can be used to aggregate controller performance across all of the runs. Based on these results, the control designer can then determine whether performance was adequate across the sampling, or if additional adjustments are needed. This process can also highlight regions of the design space that may be more critical in terms of meeting requirements, where additional information about the limiting conditions might lead to a better overall design.

II. GAS-TURBINE FUEL SYSTEM

In this study, a closed-loop controller will be developed for a gas-turbine fuel system. Turbine engines often have multiple variable geometries, each with their own set of controlling hardware, but rarely do these systems compare to the design challenges of the fuel system. Fuel control is directly tied to gas-turbine performance and operability and is often associated with engine constraints as well. The hardware models for the gas-turbine fuel system that will be used in this study are physics-based. The models are initially developed from first principles and later fine tuned once testing data become available. Details of each of the components of the closed-loop system are modeled, including nonlinear characteristics. As with any manufactured system, some degree of uncertainty is expected due to part-to-part variation and operating condition variation. These uncertainties are included in the model for controller performance and robustness analysis.

The actuation system consists of an electrohydraulic servo valve (EHSV), a single actuator, and a throttling valve also known as a fuel metering unit (FMU). The EHSV takes an electrical current signal from the controller and provides corresponding fuel flow to the actuator. The actuator slides based on the flows and adjusts the fuel port areas as a function of stroke. The throttling valve works to maintain a constant fuel pressure across the actuator fuel ports, which yields a predictable fuel flow for a given port area. This fuel flow leads to a series of manifolds and eventually to the combustor’s fuel nozzles. The actuator



position is fed back to the controller through a linear variable displacement transducer, or LVDT. Figure 12 shows a simple schematic of the closed-loop system.

In the EHSV model, a null current is subtracted from the input current from the controller, and then passes through a hysteresis model to account for current direction changes. This current then passes through the first of two first-order lags to account for current rise time. The lagged current value is input into an interpolating lookup table to determine the corresponding fuel flow and to which side of the actuator it is flowing. This lookup is linear over large regions, but does have some nonlinear effects near the zero crossing as well as the end points. The calculated flow is corrected for pressure and density conditions as compared to the rated conditions. The resulting flow passes through the second first-order lag, the output of which is the flow to the actuator.

There are a number of uncertainties associated with the EHSV model. Part-to-part variation can lead to slight differences in null current level as well as the magnitude of the hysteresis term, leading to instances of uncertainty. The time constants on the two first-order lags are also modeled with uncertainty to account for fleet variation. A gain modifier on the flow output of the EHSV is also included, centered on unity.

The actuator model is fairly simple and straightforward; the flow provided by the EHSV is divided by the actuator bore area and integrated to determine the actuator stroke position. The actuator stroke can be correlated to a fuel port area, and the resulting flow is related to the pressure across this area. Uncertainty in the actuator is tied to bore and cylinder area variations combined with frictional differences, but the levels of these are very small in comparison to some of the other uncertainties in the loop, and so they will not be considered in this study.

The throttling valve introduces additional nonlinearities and uncertainties, but these characteristics are all downstream of the closed-loop system. The actuator position is the feedback used by the control; the fuel flow output of the system is not measured. The high level closed-loop control can adjust fuel flow demands by using other sensed quantities in the gas turbine, and so the dynamics of the throttling valve will not be considered for this study on the inner-loop controller.

The sensing system model consists of a second-order lag to represent the LVDT. Uncertainty is included in this model as well, and variation is taken into account in both the natural frequency and damping terms. The output of the LVDT is a pair of

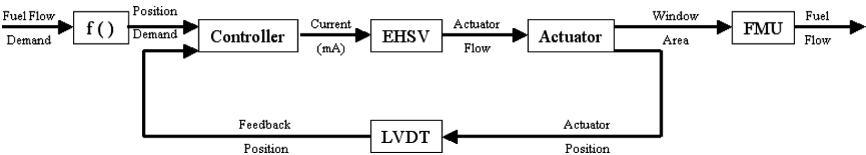


FIG. 12 Diagram of closed-loop fuel control system.

voltage signals corresponding to the position of the actuator. It is customary to model noise on these signals, but this factor was left out of the model for this study. Usually in this type of system, the controller hardware can introduce some slight dynamics to the closed-loop system. However, in many cases such as this instance, the contribution is small compared to the other components in the system. As a result, the dynamics from the control hardware will be neglected in this study. Also in this study, normalization will be applied to the actuator position and the fuel demands and resulting flows. This gives the results a more universal appeal without getting caught up in details of the specific application.

The current controller for this fuel system is a proportional-integral (PI) controller. The proportional component is implemented to try to give fast response times to changing demands, and the integral component is in place to try to eliminate steady-state error. Derivative control elements can be difficult to design for this type of system. As already mentioned, the voltage feedback signals are susceptible to noise, which presents a challenge for mathematical derivatives. It is possible to filter out some of the noise, but it is not a trivial task and can reduce the effectiveness of the derivative component. For the current system, control can be achieved without this component.

The current controller doesn't use fixed gains for proportional and integral components. Instead, the controller employs a technique known as gain scheduling. With this approach, the constant gains are replaced with arrays, selecting different gains based on current conditions. This essentially transforms the output space of the PI controller from a single plane to a series of planes, enabling for better optimization for the control surface. The tuning for this approach can be a challenge, having to determine the appropriate gains for each of the different operating conditions.

The gain scheduling technique shares some similarities with fuzzy logic controllers in the fact that it changes controller outputs based on input conditions, mirroring the decision-making behavior of intelligent controls. A fuzzy logic controller is inherently nonlinear, providing the optimization space similar to the gain scheduling technique, but perhaps in a more straightforward manner. One of the goals of this study is to try to evaluate a unique tuning approach for fuzzy logic controllers to try to take some of the burden off of the designer.

III. GENETIC FUZZY GAS-TURBINE FUEL SYSTEM

A fuzzy logic controller (FLC) will be utilized to provide closed-loop control of the gas-turbine fuel system in this study. As mentioned, a genetic algorithm will be used to aid in tuning the membership functions of the controller, forming a member of the generic class of systems known as genetic fuzzy systems. The fuzzy logic controller will be based on the structure proposed by Zilouchian et al. [8] in their closed-loop controller for a gas-turbine combustor pressure simulation system.

A. CONTROLLER DESIGN

The FLC used in that study consisted of two inputs and one output. The inputs consisted of the error, the difference between the desired set point and the current point, and the delta-error, or the difference between the current error and the previous. The output of the system was the control action, tied to positions for pressurizing and depressurizing valves. Here the output will be associated with the current supplied to the EHSV of the actuation system.

Each of the inputs and output will be normalized on a $[-1\ 1]$ scale, with scalar multipliers up and downstream of the fuzzy controller to reach the desired range. For the inputs, the error will be normalized by the range of the actuator, and the delta-error will follow suit. On the output, the control action will be scaled by the maximum torque motor current that can be output, covering the full output range. This scaling allows for simplification within the fuzzy membership functions while still providing coverage throughout the operating envelope. Also, by using normalized membership functions, the fuzzy controller has a more universal appeal to transfer for use in other applications.

In the study, Zilouchian et al. [8] used seven membership functions for both the inputs and the output. This study will also use seven membership functions for each parameter, although their shapes will be set a little differently. The linguistic variables that will be used are shown next, along with Fig. 13 showing membership functions for identically shaped isosceles triangles. This distribution will be used for comparative purposes.

NL: Negative Large

NM: Negative Medium

NS: Negative Small

ZE: Zero

PS: Positive Small

PM: Positive Medium

PL: Positive Large

The membership functions themselves will be triangles, although they will not necessarily be isosceles triangles as in the study by Zilouchian et al. [8]. The triangles will be nonisosceles, with the exception of the zero fuzzy set (ZE) as the fuzzy sets will be symmetric about the zero point of the axis. Using a similar approach as Nelson and Lakany demonstrate, with a few assumptions, the membership functions can be specified by a reduced number of parameters [7]. By assuming symmetry, and taking advantage of the normalization on the inputs and output, the entire set of seven membership functions can be fully specified with just two parameters. Figure 14 demonstrates how two parameters, x_1 and x_2 , can fully specify the seven nonisosceles membership functions. In this figure, x_1 had a value of 0.55, and x_2 had a value of 0.82.

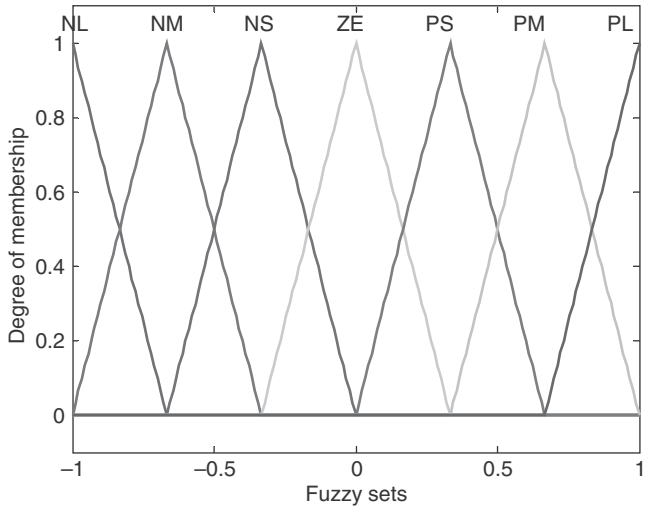


FIG. 13 Symmetric fuzzy membership functions.

With the structure shown in Fig. 14, a maximum of two fuzzy sets will be active at any time for each of the inputs, which reduces complexity of the input-to-output relationship as well as computational requirements. This method will also ensure that the sum of membership across all sets is unity, a measure of consistency. With the membership functions in place, a rule base is

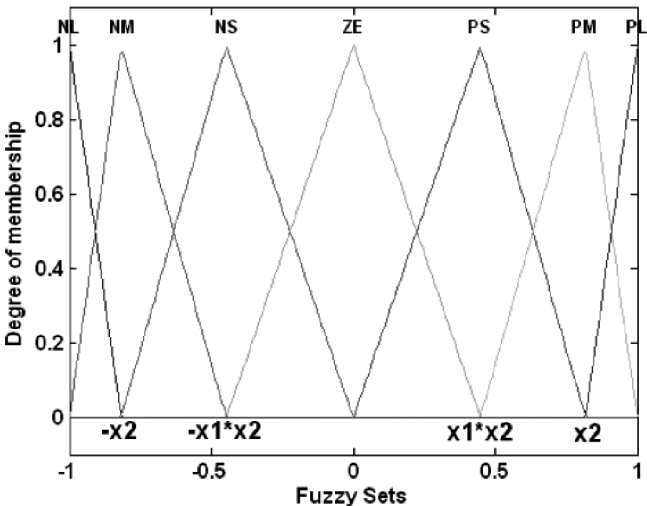


FIG. 14 Nonisosceles fuzzy membership function specification.

TABLE 3 FUZZY LOGIC CONTROLLER RULE BASE FOR SEVEN MEMBERSHIP FUNCTIONS [8]

		Delta-Error						
		NL	NM	NS	ZE	PS	PM	PL
Error	NL	NL	NL	NL	NL	NM	NS	ZE
	NM	NL	NL	NL	NM	NS	ZE	PS
	NS	NL	NL	NM	NS	ZE	PS	PM
	ZE	NL	NM	NS	ZE	PS	PM	PL
	PS	NM	NS	ZE	PS	PM	PL	PL
	PM	NS	ZE	PS	PM	PL	PL	PL
	PL	ZE	PS	PM	PL	PL	PL	PL

needed (see Table 3). The set proposed by Zilouchian et al. will be used here [8]. The rule base consists of 49 IF-THEN rules, forming a complete set for the two inputs with seven fuzzy sets each. The rule base is illustrated in Table 1. As an example, the first rule would read, “IF ERROR is *NL* AND DELTA-ERROR is *NL*, THEN CURRENT is *NL*.”

Using the seven membership functions for the fuzzy output sets, the last piece of the fuzzy controller needed is the defuzzification method. With the selected membership functions and rule base, the fuzzy sets of the output will always be adjacent. As a result, the centroid defuzzification method can be used, as this fuzzy logic controller will not encounter some of the shortcomings that prevent this simple method from being utilized in other contexts.

With the details of the fuzzy logic controller in place, the representation of these parameters in a form the genetic algorithm can act upon is needed. As mentioned earlier, both of the inputs and the output will each have two parameters defining their membership functions, leading to a total of six parameters to be used for tuning (see Table 4). Simple binary encoding will be used, and because

TABLE 4 SAMPLE FUZZY MEMBERSHIP FUNCTION CHARACTERISTIC PARAMETERS

Parameter	String	Value
1	01100011	0.3882
2	00010010	0.0706
3	01111100	0.4863
4	01100011	0.3882
5	11101011	0.9216
6	11100111	0.9059

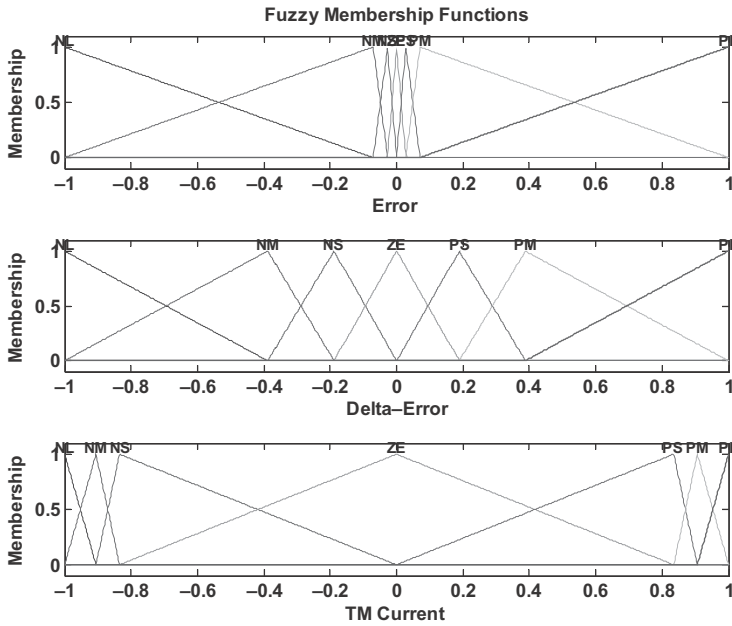


FIG. 15 Sample nonisocles fuzzy membership functions.

of the reduced number of parameters to be represented, more bits can be used to provide more granularity for optimization. Eight bits will be used for each parameter, leaving each chromosome to be encoded with 48 bits total. The first two sets of eight bits will be used for input one (ERROR), the second set for input two (DELTA-ERROR), and the last pair will be used for the output (CURRENT) (see Fig. 15). The two parameters for each of the input/output will be determined using the following relation:

$$y = y_{\min} + \frac{y_{\max} - y_{\min}}{2^n - 1} \cdot \sum_{i=0}^{n-1} s_i \cdot 2^i \tag{1}$$

Taking advantage of the normalization of the inputs and output, the determining parameters will be bounded between [0 1], giving the minimum and maximum limits. Each set of eight bits will be passed through this formula to determine the governing parameters for the fuzzy sets. An example is illustrated next.

Sample chromosome:

01100011 00010010 01111100 01100011 11101011 11100111

First parameter string:

0 1 1 0 0 0 1 1

First parameter determination:

$$y = \frac{1}{2^8 - 1} \cdot \sum_{i=0}^7 s_i \cdot 2^i$$

$$y = \frac{1}{2^8 - 1} \cdot (1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 + 0 \cdot 2^4 + 1 \cdot 2^5 + 1 \cdot 2^6 + 0 \cdot 2^7)$$

$$y = \frac{99}{255} = 0.3882 \quad (2)$$

After the fuzzy sets have been developed, the closed-loop system was simulated to determine the closed-loop response. This process was then repeated for each member of the population to assess performance. For the genetic algorithm to optimize, it must have a mathematical measure to assess the performance of each member of the population. In control system design, performance is often measured against several different criteria, each important in some aspect. The designer tries to meet various requirements, often trading performance against one criterion for another, trying to develop an overall balanced design. This system isn't any different, requiring a design that tries to maximize performance against several different criteria.

In this study, the performance of the closed-loop system to a step input will be assessed using an aggregate of the classical measures of rise time (Trise), settling time (Tset - 2%), percent overshoot (%OS), and percent steady-state error (%SSE). It is necessary to combine these measures into a simple, comprehensive form on which the genetic algorithm can optimize. For this study they will be summed, but first they need to be nondimensionalized in some manner and placed on the same order of magnitude. For this task, each of the classical measures will be divided by the respective performance measure of a fuzzy logic controller with identically shaped isosceles triangles, resulting in a percent comparison to this baseline configuration. The sum will then be utilized by the genetic algorithm for optimization. It is common practice to use these parameters in the design process as measures of controller performance, but oftentimes some measures are more important overall than others. In this study, in addition to normalization, these parameters will have weightings applied within the fitness function to try to reflect priorities in the design with respect to these parameters.

In this design, it was determined that steady-state error was the most important of these measures, as accurate fuel control is vital to gas-turbine performance. Without accurate fuel control, the engine might lose efficiency or violate operating constraints, negatively impacting range or payload of aircraft, or power output by generation-based turbines. The steady-state error term in the fitness function was given a weighting of 4. Settling time was also deemed more important than the remaining two factors, but not quite to the level of steady-state error. When a given operating condition is demanded, the operator will want the turbine engine to attain that condition rapidly. An extreme example that illustrates this

importance could be seen in a combat aircraft, relying on fast and accurate throttle control providing a combat advantage. A weighting of 2 will be applied to the normalized settling time term. Rise time is an important measure for getting the engine most of the way to the desired point, but typically to a lesser degree than accuracy and settling time. A rapid response is typically desired, but fast rise times can lead to conflict in terms of other performance characteristics, as they can lead to higher overshoots and longer settling times. By incorporating all of these measures into the fitness function for the genetic algorithm, a design that is balanced with respect to these criteria can be sought.

This priority system just discussed will be reflected within the fitness function. By defining the fitness function as the sum of weighted performance measures, the genetic algorithm can work on improving system performance in a manner consistent with priorities. Using this approach, performance improvements in one area will be held in check with performance in the other categories, making sure the system won't completely sacrifice performance in one area for improvement in others. The weighting scheme on the normalized performance components is illustrated in the fitness function relation in Eq. (3). The second relation is optional, depending on how the genetic algorithm is setup (to minimize or maximize fitness function values).

$$f = \left(4 * \frac{\%SSE}{\%SSE_{bl}} + 2 * \frac{Tset}{Tset_{bl}} + \frac{Trise}{Trise_{bl}} + \frac{\%OS}{\%OS_{bl}} \right) / Wsum$$

$$Wsum = 4 + 2 + 1 + 1 = 8$$

$$F = 1/f$$
(3)

The population for the genetic algorithm will consist of 10 members, each represented with a 48-bit binary string. This number was chosen to give opportunities for optimization but to try to limit the computation of each iteration. The genetic algorithm will run through 200 iterations for each run. With any genetic algorithm, premature convergence has the potential to hinder some optimization if an early candidate performs extremely well compared to the other candidates. To try to alleviate this shortcoming, this study will perform multiple runs of the genetic algorithm, and the best solution overall will be chosen. The initial population for each trial will be developed at random, each bit having equal opportunity of being a 0 or 1. With each initial population being chosen randomly, multiple runs will also help explore the design space.

Following closed-loop simulation of each member, the fitness of each is determined. Then, the candidates are ranked according to their fitness. Using an elitist scheme, the top two performers are kept for the next population. Also, the worst performing member undergoes inversion, flipping every bit character in the string, and is placed into the next population. The remaining members are determined through a selection scheme and have opportunities for genetic variation to be introduced. The remaining members of the next population are selected at

random from the current population, with a probability proportional to their fitness. The fitness of each member is divided by the fitness sum across all members, yielding the percent chance that member is selected for the next population. This can be thought of as the roulette wheel approach.

After the members of the population have been determined, these members undergo genetic variation, crossovers, and mutations. This scheme will include both one- and two-point crossovers. A design parameter of the genetic algorithm is the rates for these crossovers. The overall chance of a member undergoing a crossover is set at 0.8, of which $2/3$ will be one-point crossovers and two-point crossovers will be performed on the residual. Using an elitist scheme enables a high crossover rate, as the best performing candidates are already preserved. The goal of crossovers is to try to recombine pieces of existing members to form new members, which may be better suited for the controller.

Following crossovers, the population (excluding elites and inverts) is subjected to mutation. Again, the mutation rate is a design parameter for the algorithm developer, and in this case was chosen as 0.02. This means that each bit in each chromosome string has a 2% chance of being inverted. Multiple mutations can occur within the same string, or some may be left without any alteration. Mutations are a way of trying to develop new or unexplored genetic material, which may move closer to an optimized solution.

Once the successive population has been selected and has had opportunities for genetic variation to be introduced, it is ready for the next iteration of the genetic algorithm. The members of the new population go through the same process, having their fitness determined from closed-loop controller performance, determining the best and worst performers, and determining the makeup of the next population. This process is repeated for each iteration, and over time a more optimized solution than the original population should be produced.

B. CONTROLLER PERFORMANCE ANALYSIS

With the binary representation, population selection, and genetic variation rates in place, the genetic algorithm is ready to be run. For this study, the algorithm was run five different times, each with a different random initial population, and the best performing candidate and its corresponding fitness rating was saved from each run. In the end, the member of the population with the best overall fitness rating was selected for the final controller design. A comparison of the best performer from each of the runs is shown in Table 5. This table shows the chromosome string from each candidate, the results for the performance measures, and the corresponding fitness value.

In general, the performance of each of the runs is pretty comparable. In fact, runs 3 and 5 produced the exact same solution for the best performing candidate. The solution found from the first run had the best overall performance, with a fitness rating of 13.01. Even though the other solutions did not quite result in the same level of performance, they all still demonstrated significant improvement

TABLE 5 GENETIC ALGORITHM PERFORMANCE COMPARISON

Run	Chromosome	Trise	Tset	%OS	%SSE	Fitness
1	100111110000101000101000000010000010110011110110	0.250	0.363	0.0000	0.0008	8.18
2	110000110000100100010000100001001000000010111110	0.225	0.325	0.0882	0.0882	6.08
3	01111111000011100001001000010010001111111110000	0.250	0.375	0.0036	0.0036	7.86
4	100100100001000000001111001000110100010011111111	0.250	0.350	0.0405	0.0405	6.94
5	011010110001000100111011100011001001000111111111	0.188	0.275	0.2114	0.1440	5.25

TABLE 6 SYMMETRIC FUZZY SYSTEM PERFORMANCE CHART

Trise	Tset	%OS	%SSE	Fitness
1.21	1.78	1.193	1.193	0.7456

over the symmetric case, whose performance parameters are shown below. The values listed in Table 6 served as the baseline case for normalizing the fitness function components. By definition, the fitness for this case is unity.

Figure 16 illustrates the performance of the genetic algorithm over the iterations of the first run, which resulted in the best performing controller to be used for future evaluation. In the first figure, the fitness rating of the best performer is shown. Because an elitist scheme was used, the fitness will gradually increase from the initial population, as new members created through genetic variation perform better than their predecessors. In the second figure, Fig. 17, the controller performance parameters are shown against the iterations of the genetic algorithm.

Using an elitist scheme ensures the best performer is kept from one iteration to the next; however, because overall performance is based on a weighted combination of four individual performance parameters, it is possible to see some increases in the minimums in the general downward trend. Details of the final controller design are shown next, starting with the binary representation found through the genetic algorithm, showing a table of the binary translation, and lastly a plot of the fuzzy sets:

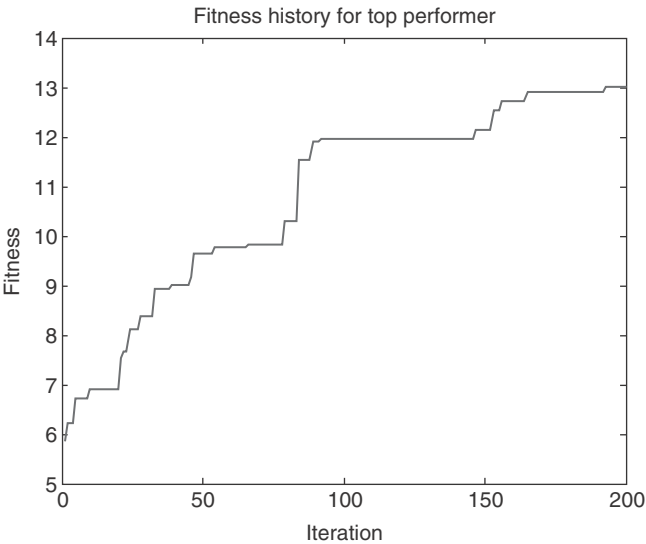


FIG. 16 Fitness function value history of selected fuzzy controller.

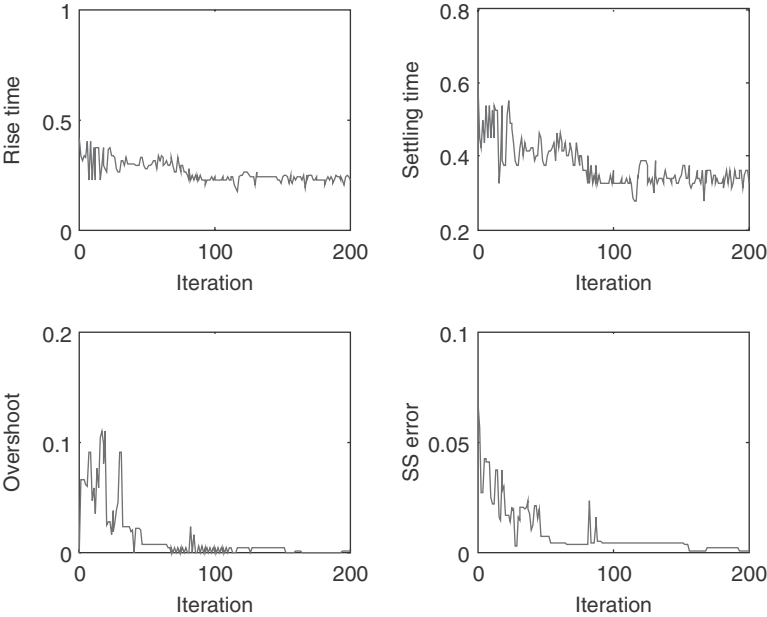


FIG. 17 Performance measure history of selected fuzzy controller.

The final design chromosome:
10011111 00001010 00101000 00001000 00101100 11110110

The final design parameter table is Table 7.
Final design fuzzy sets are shown in Fig. 18. Some conclusions can be drawn from the resulting distributions of the fuzzy sets. On the input side (error and delta-error), the membership functions are aggregated near the zero point. This

TABLE 7 SELECTED FUZZY MEMBERSHIP FUNCTION CHARACTERISTIC PARAMETERS

Parameter	String	Value
1	10011111	0.6235
2	00001010	0.0392
3	00101000	0.1569
4	00001000	0.0314
5	00101100	0.1725
6	11110110	0.9647

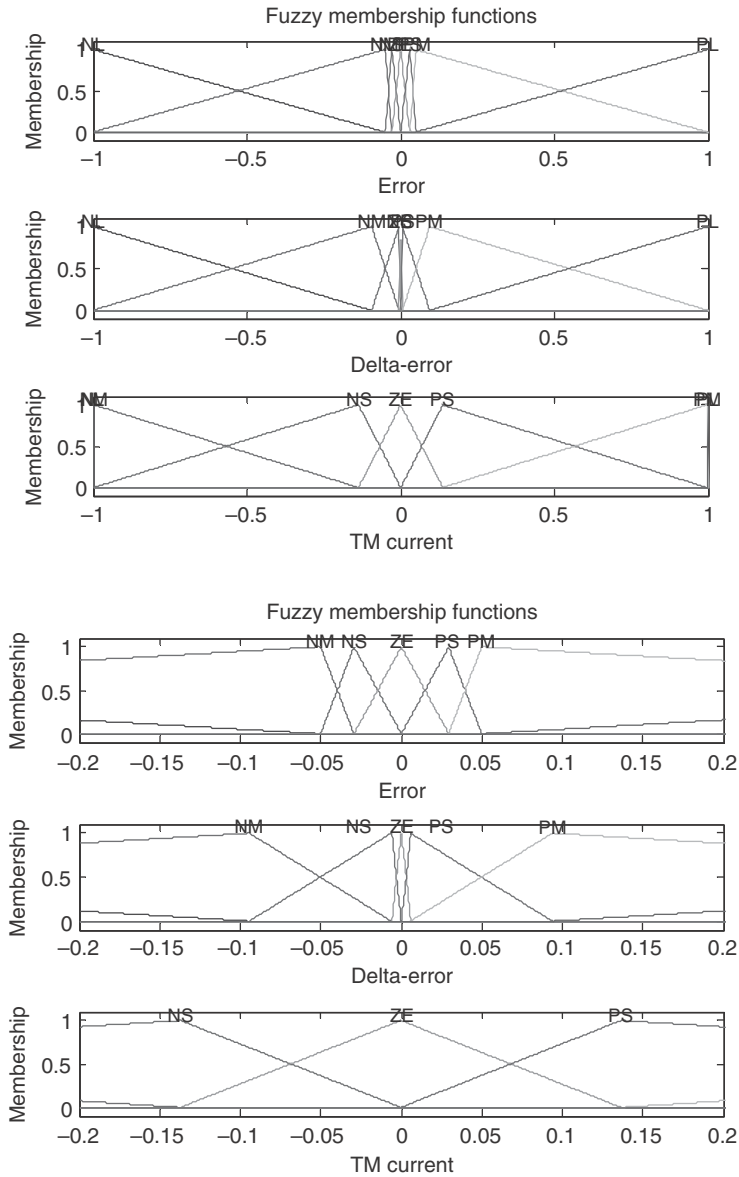


FIG. 18 Selected fuzzy membership functions.

is likely driven by the high weighting on the steady-state error term in the fitness function. By increasing the granularity in this area, the system can better respond to drive out any error resulting in the low steady-state errors seen from simulating

the system. This structure for the inputs also serves to reduce overshoot, as the system can tone down the output demands as the system gets close to the target value.

On the output side (TM current), it is interesting to note how the large classes (NL, PL) have been pushed way out towards the normalized limits. Looking back at Table 3, these categories come in to play in the upper-left and lower-right corners, where the inputs tend to be in the medium and large categories. This will result in a large controller output, leading to a fast response of the system when in a large error or delta error region. In the closed-loop system, this results in lower rise times, aggressively driving the system to smaller errors (and deltas). Figure 18 and the accompanying discussion demonstrate the compartmentalization property of fuzzy control systems. Different regions in the input space will elicit different behaviors resulting from the output space, allowing specific regions to be tuned to achieve desired performance. This specialization offered by the simple nonlinear controller demonstrates a key advantage over a traditional PI controller, whose control response could be mapped to a single plane.

Another way of visualizing the resulting fuzzy membership functions is through a surface plot, shown in Fig. 19. This plot combines inputs, the output, and the fuzzy controller rule base into a single figure that can help illustrate key characteristics of the controller. This figure reinforces visually the behavior

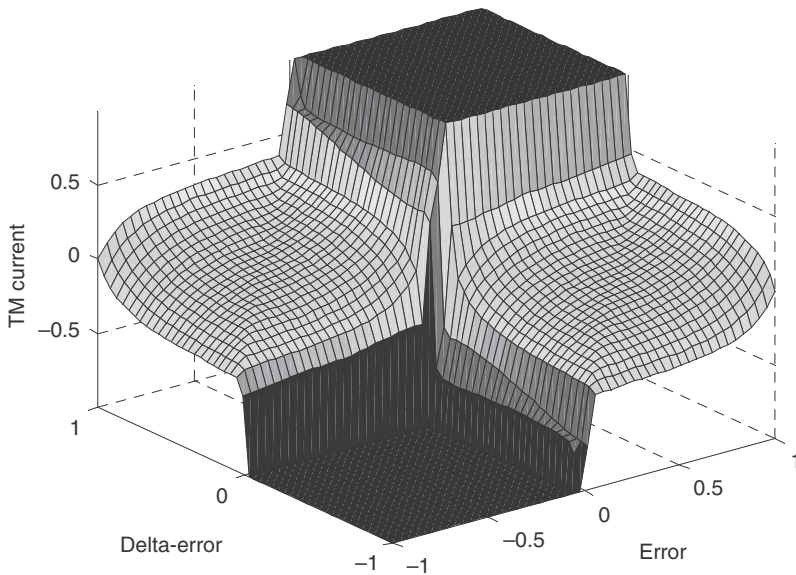


FIG. 19 Surface plot of selected fuzzy controller.

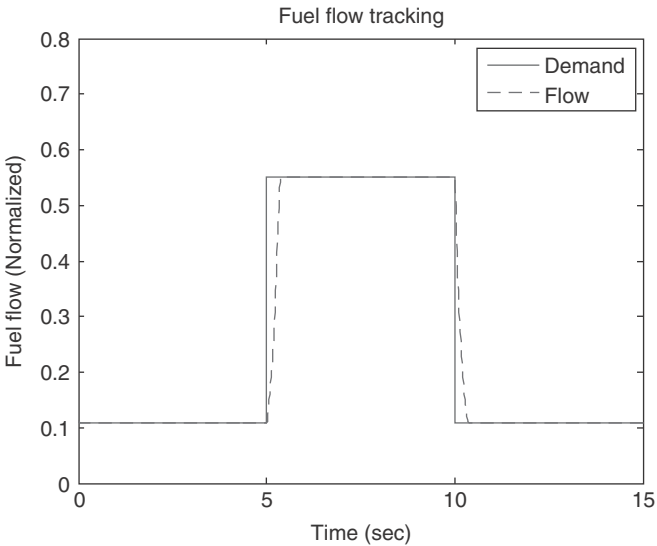


FIG. 20 Fuzzy logic controller step responses.

that large outputs will result from larger categories of the inputs when they share signs and the finer granularity in the output as the inputs approach zero.

With the membership functions for the fuzzy sets delivered by the genetic algorithm, the closed-loop system can be simulated for any mission profile or fleet sample. One of the obvious cases is a simple step response, the case for which the controller was tuned to. Figure 20 demonstrates the response to both a positive and negative step, to check for controller consistency as expected as a result of the rule base and symmetry about the zero axis in the membership functions.

To determine the benefits of the genetic algorithm, the tuned nonisosceles fuzzy controller can be compared to the symmetric case using the same rule base. This is illustrated in Fig. 21 and is a visual representation of the data illustrated in Tables 5 and 6.

Although a step input was used to tune the controller as well as for initial simulation results, this type of input may not be the only type seen by the controller in deployment. It would be important to check the response characteristics of the controller for any input that may be experience in operation. The controller response to a ramping input and a sinusoidal input are illustrated in Figs. 22 and 23. In these figures, fuel flow tracking is plotted along with the actuator position tracking to illustrate some of the nonlinear behavior. Note that while the controller was not tuned for these cases, the controller still performs very well to both inputs, showing good tracking behavior to both sets of input types.

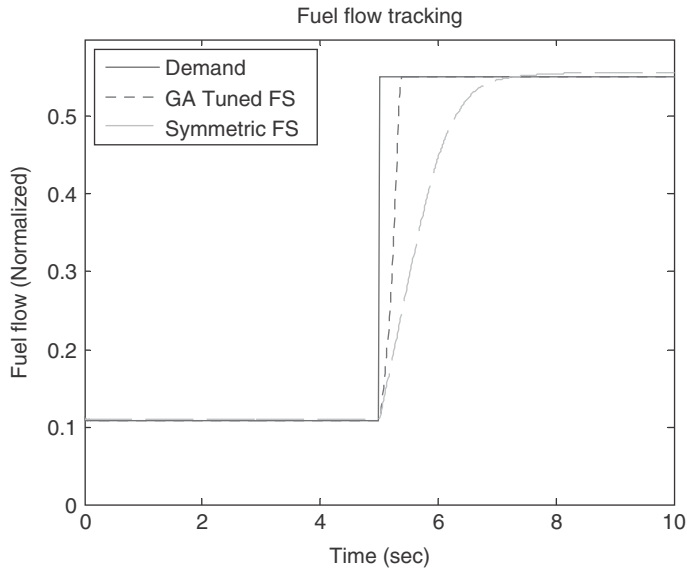


FIG. 21 Fuzzy logic controller step response comparison.

The types of inputs just shown could also be used for tuning the control with a genetic algorithm, if it was more fitting to the controller application or design requirements. If ramp tracking was more critical to the design, or if certain control bandwidth requirements were mandated by the application, then the optimization process could be turned towards such. This information would be tied to the fitness determination, and the genetic algorithm could work to optimize to these conditions. The fitness function could also combine performance against various criteria or requirements to drive towards a balanced design in a competing space.

C. ROBUSTNESS ANALYSIS

With the controller design selected in the earlier section, the next step is to gauge the performance robustness of the closed-loop system. This study will use the stochastic robustness analysis demonstrated by Stengel and others in many applications characterized by some level of uncertainty [16, 18]. This analysis involves a Monte Carlo simulation of the closed-loop system, considering variation within the components or across operating conditions. This study will consider variation within the modeled hardware components, representing part-to-part variation within a fleet or perhaps component degradation over

time. Component characteristics with uncertainty considered in this study are summarized here:

- Electrohydraulic servo valve
 - Null current
 - Hysteresis
 - Current time constant
 - Flow time constant
 - Flow gain
- Linear variable displacement transducer
 - Natural frequency
 - Damping

The uncertainty associated with each of the parameters given in the preceding list will be represented by a Gaussian distribution, with mean values and standard deviations representative of variation within each component. In this study, a

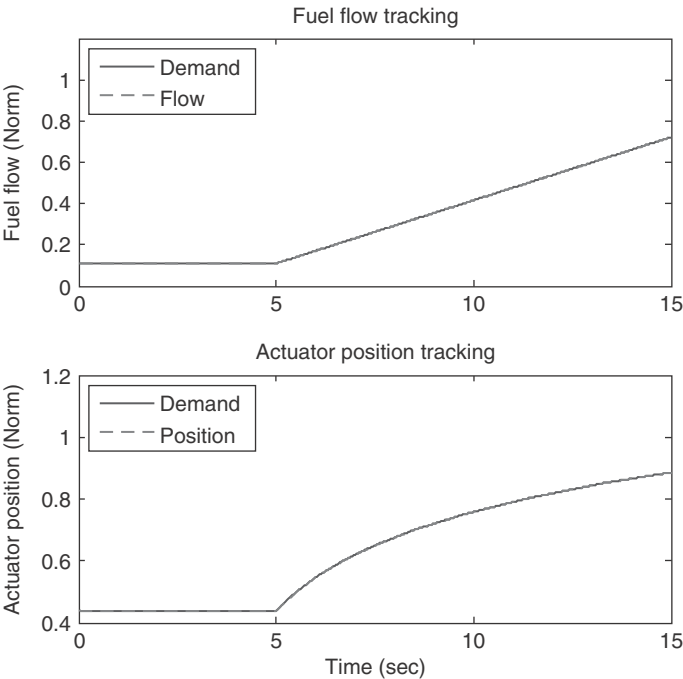


FIG. 22 Fuzzy logic controller ramp response.

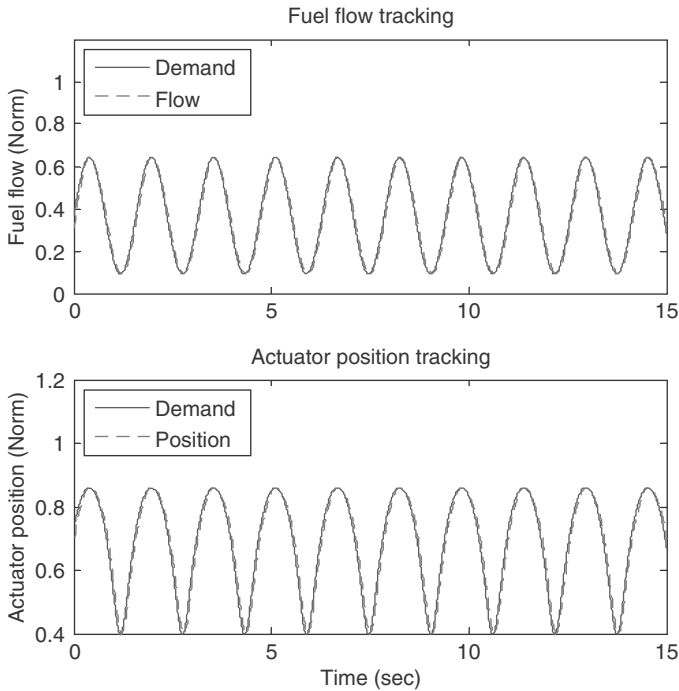


FIG. 23 Fuzzy logic controller sinusoid response.

total of 1000 closed-loop simulations will be run, each using a different set or variation parameters determined randomly from the Gaussian distributions for each uncertainty term. The system will be simulated with a step input in fuel demand, as in the case used to tune the controller. Each simulation will be gauged for performance against parameters just discussed, mainly the rise and settling times, overshoot, and steady-state error. The fitness function will also be generated for each case as means to compare performance to the design case and gauge performance across all of the Monte Carlo runs. In some cases, it may make sense for the robustness analysis to be performed on a collection of simulation cases, such as a complete mission analysis or cases known to be challenging in terms of performance. The selection is up to the designer and at a minimum should be able to assess controller performance against the design requirements.

To assess controller robustness, criteria need to be developed for determining if performance is adequate. These criteria will be selected to provide a binary determination, or pass/fail criteria. For the purposes of this study, performance will be deemed acceptable for rise and settling time requirements if the times are less than double those of the design case, or less than 0.50 and 0.725 s, respectively. In terms of overshoot, a limit of 0.5% will be used; anything more and

performance will be considered inadequate. For the steady-state error limit, a value of 0.1% will be used. The performance of each run will be compared to these limits, and the robustness of the controller will be determined based on the aggregate results across all simulations. The results of the stochastic robustness analysis are illustrated next, beginning with Fig. 24. This figure illustrates the fitness function values calculated based on the results of each of the individual simulations. The results generally hover in the 9–12 range with a few outliers. The extreme values (8.06, 13.8) are also highlighted in Fig. 24.

Another means of looking at these data is using a histogram, which basically counts the number of occurrences within a set of ranges. This is illustrated in Fig. 25. Figure 25 helps illustrate the importance of performing a robustness analysis once an iteration of the controller design is in place. The fitness function value of the selected controller was just over 13 for the design case, but as just shown, a vast majority of the simulation cases are below this value. This demonstrates how uncertainty in the loop can lead to a decrease in expected performance, and why a stochastic robustness study is important to show how the controller will perform across a fleet or set of operating conditions.

Figure 26 compares performance of the cases with the minimum and maximum fitness function values to the base case from the original design. The figure includes both a plot of the full response and a plot zoomed in at the top

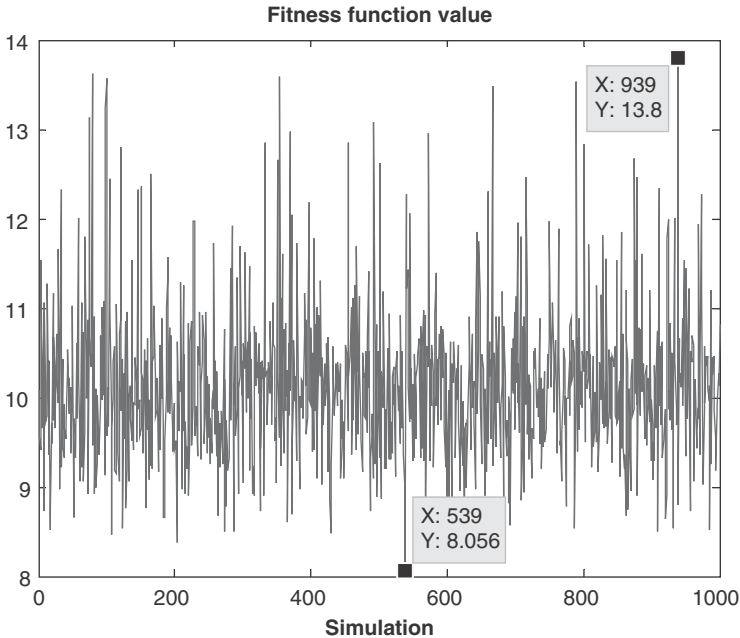


FIG. 24 Fitness Function Values for Each Monte Carlo Simulation.

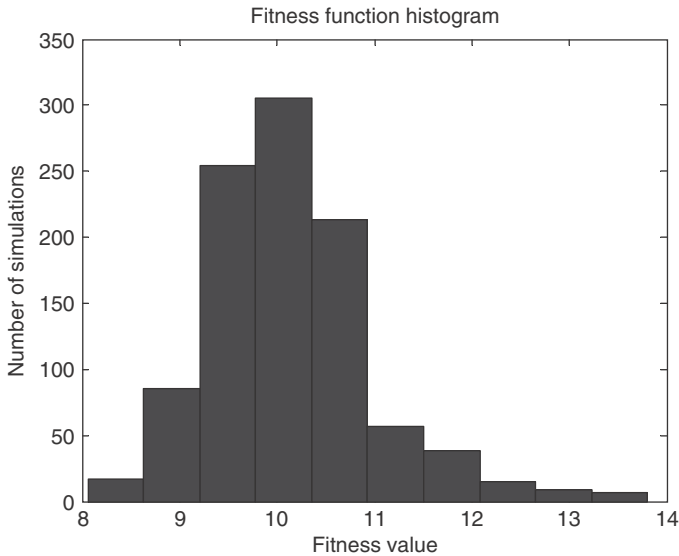


FIG. 25 Histogram of fitness function values.

of the step to capture some of the details of the compared cases. As shown, the differences are very slight, with the biggest differences seen in overshoot and steady-state error. This figure demonstrates the robustness properties of the fuzzy logic controller so that, even in a nonlinear environment with uncertainties, the controller can maintain a high level of performance.

Figure 26 offers a look at the fitness ratings from each of the simulations and some comparisons between the extreme cases. What is really important in a stochastic robustness analysis is to look at the performance measures and how well they stack up to the limits. Figure 27 shows histogram plots of the four main performance parameters across all of the runs of the Monte Carlo simulation.

Figure 27 shows each of the simulations demonstrates adequate performance against the criteria outlined earlier. Gas-turbine rotor dynamics are typically on the order of a second in terms of response time, so that even though some of the simulations show a slight decrease in performance with respect to the design case the controller still should respond fast enough to maintain adequate control. Uncertainties in a few terms of the control loop were neglected for this study. However, these terms are known to be small in comparison to the uncertainties that were considered and are not expected to affect the results to a large degree. In addition, as demonstrated in Table 8, the performance parameters have margins ranging from 30–50% to the selected limits, leaving some flexibility for these effects.

Table 8 underscores the importance of performing a robustness analysis on a controller design. Looking at the first row, the average fitness across the runs was

just over 10, a significant drop when compared to the design case of 13. As mentioned earlier, this is a strong indicator of how uncertainty can impact performance. Looking at the two rows for the timing measures, the mean case did not deviate significantly from the design case. In fact, the rise time average was actually lower than base case, albeit only by a small degree. Each of these parameters has 40% margin to the selected design limit.

Whereas the performance measures for time did not shift dramatically from the base design case, the percent overshoot and percent steady-state error measures did experience a significant change. The averages for these measures were found to be 0.06 and 0.04%, respectively, which is a relatively large change considering the design case had basically eliminated these measures. This demonstrates that the overshoot and steady-state error are more sensitive to the uncertainties associated with the closed-loop system, which again highlights the need for robustness analysis. The steady-state error reached as high as 0.07% in the Monte Carlo simulations, leaving just 30% margin to the selected design limit. As these criteria were stated to be the most important factor to the closed-loop control design, it is something to keep an eye on to evaluate carefully in future studies.

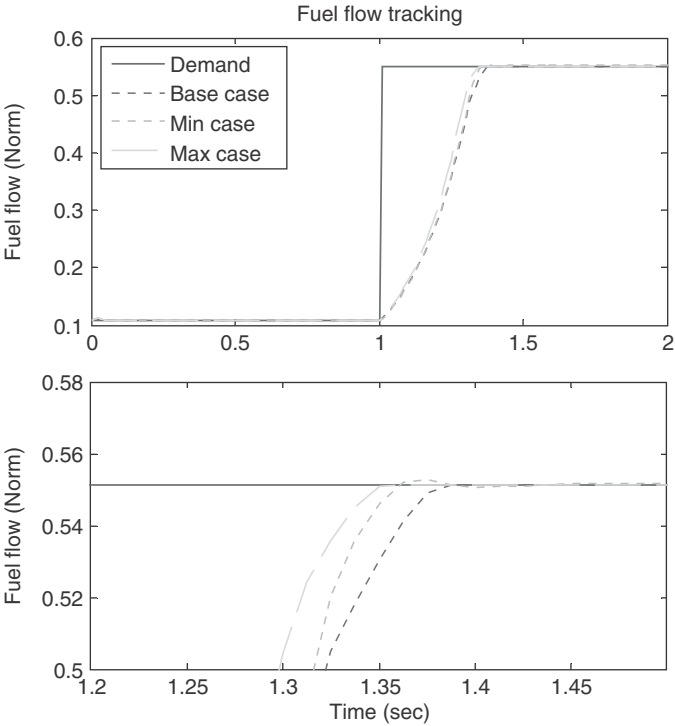


FIG. 26 Step input results comparison of minimum, maximum, and base cases.

If the control did not perform acceptably in all cases, the designer has a few options. Controller redesign is one option, depending on the severity of the violations found in the robustness analysis. The designer could examine the cases in which performance was deemed unacceptable and try to capture this information into the tuning process. After redesign, the robustness analysis would be performed again, making sure no new issues were introduced in the process, as illustrated in Fig. 11.

It is also possible to tie the stochastic robustness analysis process into the design process. Instead of having the genetic algorithm focus on a specific design case, the fitness could be based on average or minimum performance results from a robustness analysis. In this manner, the stochastic robustness analysis becomes part of the design, rather than just a check at the end to gauge performance.

Another option would be to use information from the specific performance failure cases to adjust tolerances for the system. If the stochastic analysis represented fleet variation, then an inspection process could be put in place to prevent hardware exhibiting those qualities from being deployed in the field if the component were on the fringe of the Gaussian curve. The designer could also choose to live with the performance exception, depending on the conditions and how critical to the overall control system that simulated case represents.

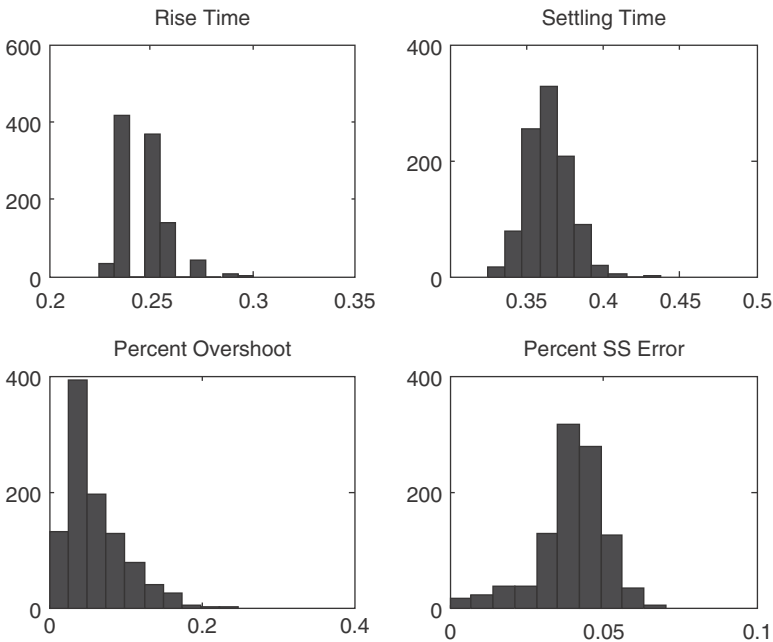


FIG. 27 Histograms of performance parameter values.

TABLE 8 STOCHASTIC ROBUSTNESS ANALYSIS PERFORMANCE SUMMARY

Parameter	Base Case	Average Case	Min Case	Max Case	Limit Case	Margin
Fitness	13.0	10.1	8.06	13.8	N/A	N/A
Rise time	0.250 s	0.247 s	0.225 s	0.300 s	0.500 s	40%
Settling time	0.363 s	0.363 s	0.325 s	0.438 s	0.726 s	40%
Overshoot	0.0002%	0.060%	0.000%	0.248%	0.500%	50%
Steady-state error	0.0003%	0.0397%	0.0001%	0.0703%	0.1000%	30%

As demonstrated, stochastic robustness analysis is a critical tool when evaluating controller design in an uncertain environment. Considering manufacturing processes, component tolerances, and operating conditions, one would be hard pressed to find a system without some level of uncertainty. This type of analysis is critical to ensuring a designer develops a robust control system, one that will be able to meet its design objectives in foreseeable environments.

IV. CONCLUSIONS

In this study, a fuzzy logic controller was designed for a gas-turbine fuel system. The design process began with the basic structure used by Zilouchian and others in their study for a fuel system simulation bench [8]. This study contributed the input/output definition, as well as the rule base. Their approach utilized identically shaped isosceles triangles for the fuzzy set definitions, which can be constraining when trying to tune the controller for improved performance. Nelson and Lakany evaluated the use of nonisosceles triangular membership functions for increased tuning flexibility, defined using an approach that reduces the number of tunable parameters to a more manageable level [7]. These two studies formed the backbone of the fuzzy logic controller utilized in this study, while a genetic algorithm was utilized in a unique tuning approach.

The genetic algorithm was used with simple binary encoding for each of the parameters governing the fuzzy sets for the inputs and output of the controller. Multiple runs of the genetic algorithm, each with a unique random initial population, were employed to try to work around the potential for premature convergence. As Table 5 illustrates, each of the genetic algorithm runs converged to a similar solution, speaking to the robustness of the algorithm itself. As the simulation results indicate, each of the solutions demonstrated significant performance improvements over the symmetric isosceles fuzzy set case. This stands to show that even with the simplifications put in place to reduce the number of tuning parameters, enough flexibility was maintained to allow for performance

TABLE 9 FUZZY LOGIC CONTROLLER COMPARISON

	Trise	Tset	%OS	%SSE	Fitness
GA tuned nonisosceles	0.250	0.363	0.0002	0.0003	13.01
Identical isosceles (baseline)	1.21	1.78	1.19	1.19	1.00
Percentage of baseline, %	21	20	0.02	0.03	—

improvements. Looking at Table 9, the rise time and settling time for the genetic-algorithm-tuned case are roughly 20% of what they were in the baseline case. The operator will experience marked improvement in the response of the gas turbine, in both the time it takes to get close to the operating point and the time it takes to settle there. Looking at the percent overshoot and steady-state error, the results show an even greater improvement. Both the overshoot and error were practically eliminated from the closed-loop system. The improvements in each of these performance measures resulted in a fitness function of 13 as compared to the baseline case of 1.

These improvements were merely due to a change in the fuzzy membership function definitions, the inputs, outputs, and rule base remained the same between the two cases. This illustrates just how important a role the membership functions play in overall controller performance and how powerful a design handle they can be. In this study, isosceles triangular membership functions were switched to nonisosceles functions. It may be possible to find an even more optimal solution by employing a different membership function structure, such as a Gaussian distribution or some custom function.

The performance of the fuzzy logic controller was improved in several measures based on the fitness function developed in this effort. The fitness function was normalized, weighted, and inverted to yield a relation that could be maximized by the genetic algorithm, where successive populations could be selected proportional to fitness. With the flexibility of genetic algorithms, there are a number of different approaches that can be used. Instead of inverting the fitness function, the algorithm could be set up to minimize the performance function if a different population selection scheme was used. Another alternative would be to multiply the fitness function by -1 instead of inverting and set up the algorithm to maximize $-f$. Either of these approaches would change the appearance of the fitness history shown in Fig. 16, where the fitness jumps up in large segments due to incremental changes in the performance characteristics because of the inversion.

A few other design decisions that could also influence the results were made regarding the genetic algorithm. Utilizing multiple runs for this study was designed to avoid the risk of premature convergence. There are a few other techniques that could be explored to try to accomplish this as well, such as using a different representation of the design parameters. The population size for the runs was also set to 10, which is on the smaller side compared to some genetic

algorithm runs. This was done to try to balance computational iterations but still give a chance to optimize. Of course, the population could be expanded to a much larger base, and perhaps multiple runs wouldn't be needed. The algorithm could have also been set up to run for a longer number of iterations, but this study tried to strike a balance between computational time and incrementally small improvements to the loop system.

Inversion was a variation process introduced into the genetic algorithm, with mixed results. In several cases, the inverted chromosome ended up performing somewhere in the middle of the population in the subsequent generation, but only in one instance did the inverted chromosome end up being a top performer in the next generation. This one case was early in the iteration process and didn't end up being the final chromosome selected, but may have accelerated the process to converging to the solution by introducing a string of good genetic material. One trend to note was that this process was less effective in later iterations as the population collectively improved overall, and inversions generally led to poorer performance in successive iterations.

With a controller design in place, a stochastic robustness analysis was performed using several simulations of the closed-loop system to determine how the controller would behave in a range of environments resulting from uncertainty in the closed-loop system. This type of analysis can be more useful than classical measures such as gain and phase margins, which can be misrepresentative in stochastic environments. This analysis only used a single test case for illustrative purposes, but could be expanded for a more complete mission analysis. The results of the analysis here showed the controller performed adequately in each simulation of the Monte Carlo analysis, with little degradation when variation was considered. This illustrates the inherent robustness properties of fuzzy logic controllers and their abilities to operate in varying environments.

The fuzzy logic controller was shown to be a viable option for the given environment, and the genetic algorithm proved to be very beneficial in the tuning process. The algorithm did take a bit of computational time considering each iteration of each run; however, once a streamlined process is set up, computational time can be much more efficient than a manual tuning process. Although the tuning case and fitness functions were pretty straightforward here, additional cases or complexities could be molded into the process using the same tool set outlined here. This study demonstrated the viability of genetic fuzzy systems for hardware control, a partnership that can be used in future control system design projects in the push for more intelligent systems.

REFERENCES

- [1] Zadeh, L. A., "Fuzzy Sets," *Information and Control*, Vol. 8, No. 3, 1965, pp. 338–353.
- [2] Mamdani, E. H., "Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis," *IEEE Transactions on Computers*, Vol. 26, No. 12, 1977, pp. 1182–1191.

- [3] Cordon, O., Herrera, F., Hoffmann, F., and Magdalena, L., *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, World Scientific, NJ, 2001.
- [4] Mendel, J. M., "Fuzzy Logic Systems for Engineering: A Tutorial," *Proceedings of the IEEE*, Vol. 83, No. 3, 1995, pp. 345–377.
- [5] Stengel, R. F., Ray, L. R., and Marrison, C. I., "Probabilistic Evaluation of Control System Robustness," *International Journal of Systems Science*, Vol. 26, No. 7, 1995, pp. 1363–1382.
- [6] Cohen, K., Weller, T., and Ben Asher, J. Z., "Control of Linear Second-Order Systems by Fuzzy Logic-Based Algorithm," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 3, 2001, pp. 494–501.
- [7] Nelson, G. M., and Lakany, H., "An Investigation into the Application of Fuzzy Logic Control to Industrial Gas Turbines," *Journal of Engineering for Gas Turbines and Power, Transactions of the ASME*, Vol. 129, No. 4, 2007, pp. 1138–1142.
- [8] Zilouchian, A., Juliano, M., Healy, T., and Davis, J., "Design of a Fuzzy Logic Controller for a Jet Engine Fuel System," *Control Engineering Practice*, Vol. 8, No. 8, 2000, pp. 873–883.
- [9] Holland, J. H., *Adaptation in Natural and Artificial Systems*. Univ. of Michigan Press, Ann Arbor, MI, 1975.
- [10] Nyongesa, H. O., "Enhancing Neural Control Systems by Fuzzy Logic and Evolutionary Reinforcement," *Neural Computing and Applications*, Vol. 7, No. 2, 1998, pp. 121–130.
- [11] Radziszewski, L., and Kekez, M., "Application of a Genetic-Fuzzy System to Diesel Engine Pressure Modeling," *The International Journal of Advanced Manufacturing Technology*, Vol. 46, No. 1–4, 2009, pp. 1–9.
- [12] Stengel, R. F., and Marrison, C. I., "Robustness of Solutions to a Benchmark Control Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 5, 1992, pp. 1060–1067.
- [13] Ray, L. R., and Stengel, R. F., "Application of Stochastic Robustness to Aircraft Control Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 6, 1991, pp. 1251–1259.
- [14] Ray, L. R., and Stengel, R. F., "Computer-Sided Analysis of Linear Control System Robustness," *Mechatronics*, Vol. 3, No. 1, 1993, pp. 119–124.
- [15] Stengel, R. F., "Toward Intelligent Flight Control," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 6, 1993, pp. 1699–1717.
- [16] Stengel, R. F., and Ray, L. R., "Stochastic Robustness of Linear Time-Invariant Control Systems," *IEEE Transactions on Automatic Control*, Vol. 36, No. 1, 1991, pp. 82–87.
- [17] Vick, A., and Cohen, K., "Longitudinal Stability Augmentation Using a Fuzzy Logic Based PID Controller," North American Fuzzy Information Processing Society Annual Conference, NAFIPS-2009-5156402, Cincinnati, OH, June 2009.
- [18] Wang, Q., and Stengel, R. F., "Robust Control of Nonlinear Systems with Parametric Uncertainty," *Automatica*, Vol. 38, No. 9, 2002, pp. 1591–1599.

Multiresolution State-Space Discretization Method for Q-Learning for One or More Regions of Interest

Amanda Lampton* and John Valasek†

Texas A&M University, College Station, Texas 77843-3141

I. INTRODUCTION

For the computational reinforcement learning problem, discretizing the state and action spaces is a common way to cast a continuous state and action space problem as a reinforcement learning problem. A simple learning problem can be easily discretized into a relatively small number of states. The learned value or action-value function is generally a good representation of the agent's knowledge of the environment. A problem becomes more complex as the number of state variables needed to represent the environment increases. The number of states in the action-value function depends on how a problem is discretized. There is a tradeoff, however. If the agent can only store knowledge in a small number of states, important details of the environment may be lost. If the agent can store knowledge in a very large number of states, details of the environment are captured quite well. The caveat is that the rate of convergence drops drastically as the number of states increases.

Hierarchical reinforcement learning (HRL) seeks to split a complex learning problem into a hierarchy of subtasks or subgoals that can be learned individually. Determining those subgoals can prove to be a challenge. This type of learning is similar in thought to the multiresolution state-space discretization method developed in this paper. Reference [1] discusses an algorithm that combines Q-learning and a locally weighted learning method to select behavioral primitives and generate subgoals for the agent. Reference [2] identifies subgoals by partitioning local state transition graphs. Reference [3] develops the theory of quad-Q-learning. This method follows the “divide-and-conquer” mentality. Not strictly HRL, but the general thought behind the method is similar. The algorithm treats state transitions slightly differently than in traditional Q-learning. Rather than being in a

*Postdoctoral Research Assistant, Texas A & M University; currently Staff Engineer, Research, Systems Technology, Inc., Hawthorne, CA; alampton@gmail.com. Student Member AIAA.

†Associate Professor and Director, Vehicle Systems & Control Laboratory, Aerospace Engineering Department; valasek@tamu.edu. Associate Fellow AIAA.

state, choosing an action, and transitioning to some new state, when an action is chosen in quad-Q-learning, either there is a reward and no transition, or there is no reward, and four new states result. Each new state is treated as a new environment, and learning commences in a similar fashion.

MAXQ is a popular HRL method that decomposes a learning problem into a hierarchy of subtasks to be learned using Q-learning. Thus, it is a hierarchical Q-learning algorithm. The method is first developed by Dietterich in Ref. [4]. This paper defines the algorithm, proves convergence, and empirically shows that it learns faster the Q-learning alone. This algorithm is further developed into a model-free learning algorithm MAXQ-Q, which is also shown to learn faster than Q-learning [5]. Mehta et al. [6] develop an algorithm that works in conjunction with MAXQ. This algorithm discovers MAXQ task hierarchies and subtasks by analyzing the relationships among actions. References [7–9] also contain MAXQ in their frameworks. Reference [8] integrates MAXQ with genetic programming, which explores possible hierarchies. Multiagent reinforcement learning and HRL are combined by Makar et al. with promising results [9].

Kirchner applies a hierarchical form of Q-learning to a six-legged walking machine [10]. Movement is split into the elementary swing and stance movements of individual legs and the overall coordination scheme to achieve forward movement. The highest layer uses these learned movements to achieve goals in the environment. Even the elementary swing and stance movement of the leg is split into four separate parts to reduce problem complexity. Reference [11] uses hierarchical Q-learning for local robot navigation. Reference [12] describes a different application of HRL in which Ghavamzadeh and Mahadevan develop a multiagent HRL algorithm that learns how to optimize necessary communication between agents such that they coordinate movements to achieve goals. Reference [13] uses HRL to represent behavior for efficient creative searches to mimic the evolution of human creative behavior in intelligent systems.

A multiresolution state-space discretization method for Q-learning can improve the algorithm by gathering the most detailed information in and around regions of interest, namely, the goal. Q-learning on a continuous domain quickly becomes intractable when one considers that convergence of the algorithm to the optimal action-value function is only guaranteed if the agent visits every possible state an infinite number of times [14]. An agent would therefore visit an infinite number of states using an infinite number of actions an infinite number of times. Add in the fact that the states can be defined by an arbitrary finite number continuous variables, and the dimensionality of the problem becomes a significant issue. This multiresolution method provides a means of exploring state-action pairs (s, a) and learning the action-value function $Q^\pi(s, a)$ for a fixed policy π in progressively finer detail. It seeks a compromise between the high rate of convergence of a coarse discretization, with the high level of detail of a fine discretization. It is desired that the method be

extensible from one to N dimensions and be easily extended to other reinforcement learning problems similar to those presented in this paper.

Generally, the purpose of the multiresolution methods in machine learning is to refine the value function $V(s)$ for state s , where needed, such as when the agent finds itself in a difficult situation and the current resolution of the value function does not offer an escape. Finer resolutions could yield a path out of the situation. This is not the case with the method presented in this paper. The multiresolution state-space discretization method is driven instead by identified regions of interest. Once identified, the method refines its search criteria and the state-space discretization iteratively until the agent can find a very specific goal and the path to it. This method essentially creates a series of smaller problems, similar to what is done in hierarchical reinforcement learning, centered around a region of interest.

Conceptually, this method can be described by a simple example. Consider a scout that is sent out initially to locate any enemy encampments. When the scout encounters an encampment, the scout maps out the perimeter and determines the best approach and retreat. The scout then has orders to penetrate the encampment, find the area housing the officers, and map the route to that area. The final refinement of the scout's orders is to locate and map a route to a particular tent. The scout has a series of progressively more specific orders that are carried out sequentially in a manner similar to this method.

The method mimics the natural tendency of people and animals to learn the broader goal before focusing in on more specific goals within the same space. This multiresolution method was first applied in [15] to the morphing airfoil architecture developed in [16–18]. This paper proposes a multiresolution state-space discretization method that investigates multiple regions of interest. The problem of learning an updraft field by an autonomous soaring uninhabited air vehicle (UAV) is cast as a reinforcement learning problem and tasked with learning the commands that produce the actions to move from some position to the middle of an updraft based on the vertical velocity of the surrounding air. The levels of discretization of the state space must be tuned such that good convergence and attention to detail are achieved. The contribution of this paper is to develop a new discretization method that allows the learning to converge quickly while still maintaining a high level of detail around *multiple* regions of interest in the environment.

This paper is organized as follows. Section II describes the mechanics of reinforcement learning and how it is implemented in Q-learning in particular. The algorithm used to solve the reinforcement learning problem learns the optimality relations between the updraft requirements and the position. The agent can then be subjected to the learned updraft field and use the relations learned to choose a good series of actions to move to the nearest updraft. The method used to discretize a continuous learning domain is developed. Sections III and IV describe the discretization method and the multiresolution state-space discretization method, respectively. Section V describes how

multiple goal regions (i.e., updrafts) are discretized in this manner and then subjected to learning. Section VI defines the policy comparison stopping criteria used to determine when either convergence is reached or additional learning is no longer needed. Section VII shows how these various components work together by applying them to the benchmark inverted pendulum problem. Section VIII briefly describes the updraft model used by the reinforcement learning agent. This section also describes how the thermal model and the reinforcement learning agent are tied together. Section VIII.C takes the fully developed multiresolution state-space discretization method, applies it to the thermal location problem for autonomous UAVs, and interprets a numerical example generated from it. Finally, conclusions are drawn from the numerical example in Sec. IX.

II. REINFORCEMENT LEARNING

Reinforcement learning is learning through interaction with the environment to achieve a goal. More specifically, it is learning to map situations to actions to maximize some numerical reward. The learner or decision-maker is the *agent* and does not know what actions to take *a priori* as is common in most forms of machine learning. Everything outside of the agent comprises the *environment*. The agent's task is to learn a policy or control strategy for choosing actions that achieves its goals. To learn the correct policy, which is a state to action mapping, $\pi : S \rightarrow A$, the agent receives a *reward*, or reinforcement, from the environment. This section summarizes reinforcement learning as laid out in [19].

The agent and environment interact continually in a series of discrete time steps, $t \in \{0, 1, 2, 3, \dots\}$. At each time step t , a series of events occur. The agent first receives some representation of the environment's state $s_t \in S$, where S is the set of all possible environment states. Based on the state, the agent chooses an action $a_t \in A(s_t)$, where $A(s_t)$ is the set of actions available to the agent in state s_t . At the next time step, the agent receives a numerical reward $r_{t+1} \in \mathbb{R}$ and is in a new state s_{t+1} .

As the agent moves from state to state selecting actions and receiving rewards, it generates a mapping, as stated earlier, of states to probabilities of selecting each possible action. This policy $\pi_t(s, a)$ is the probability that $a_t = a$ at $s_t = s$.

The agent seeks to maximize the reward it receives, or more formally, its expected return R_t . The simplest form of R_t is the sum of the rewards received after time t as shown in Eq. (1):

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \dots + r_T \quad (1)$$

where T is the final time step, assuming there is a final step. This breakdown of a sequence into a finite number of steps is called an *episode*. Discounted return denotes the sum of discounted rewards the agent tries to maximize,

Eq. (2):

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2)$$

where γ is the discount rate and is $0 \leq \gamma \leq 1$. The discount rate effectively modulates the importance of future expected rewards. If $\gamma = 0$, the agent seeks only to maximize immediate rewards. As γ approaches 1, the agent takes future rewards into account more strongly.

In this paper and for many reinforcement learning problems, it is assumed that the problems can be modeled as Markov decision processes (MDPs) and cast in the reinforcement learning problem framework. An MDP satisfies the following conditional probability distribution function:

$$\begin{aligned} \Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \dots, s_0, a_0\} \\ = \Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\} \end{aligned} \quad (3)$$

for all s_1, \dots, s_{t+1} and for all integers $t > 0$ [20]. This means that rather than the transition to state s' and receiving reward r depending on all past states, actions, and rewards, the transition to state s' and receiving reward r is *only* dependent on s_t and a_t . A problem is considered an MDP if all of the information necessary for the agent to make a decision is incorporated in the current state. The decision is not based on any past states visited and is therefore path independent.

An underlying theme of almost all algorithms used to solve reinforcement learning problems is estimating *value functions*. A value function is a function of the state or of state-action pairs that estimates how good it is, in terms of future rewards, for the agent to be in a given state. The value of a state s under some policy π is denoted $V^\pi(s)$ and is the expected return of starting in s and following π for all subsequent steps. This expectation is formalized in Eq. (4):

$$V^\pi(s) = E_\pi\{R_t | s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\} \quad (4)$$

A similar relationship exists for action-value functions $Q^\pi(s, a)$, which are defined as the value of taking action a in state s under policy π . This relationship is shown in Eq. (5):

$$Q^\pi(s, a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} \quad (5)$$

These can be estimated from experience and usually satisfy some recursive relationship. This relationship for the value function is derived in Eq. (6) and

holds for any policy π and any state s .

$$\begin{aligned}
 V^\pi(s) &= E_\pi\{R_t | s_t = s\} \\
 &= E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\} \\
 &= E_\pi\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\right\} \\
 &= \sum_a \pi(s, a) \sum_{s'} \mathcal{P}_{ss'}^a \left[\mathcal{R}_{ss'}^a + \gamma E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\right\} \right] \\
 &= \sum_a \pi(s, a) \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^\pi(s')] \tag{6}
 \end{aligned}$$

where $\mathcal{P}_{ss'}^a$ is the probability of transition from state s to state s' under action a , and $\mathcal{R}_{ss'}^a$ is the expected immediate reward on transition from s to s' under action a . Equation (6) is referred to as the Bellman equation for V^π . The Bellman equation for Q^π is

$$\begin{aligned}
 Q^\pi(s, a) &= E_\pi\{R_t | s_t = s, a_t = a\} \\
 &= E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} \\
 &= E_\pi\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a\right\} \\
 &= \sum_{s'} \mathcal{P}_{ss'}^a \left[\mathcal{R}_{ss'}^a + \gamma \sum_{a'} \pi(s', a') E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s', a_{t+1} = a'\right\} \right] \\
 &= \sum_{s'} \mathcal{P}_{ss'}^a \left[\mathcal{R}_{ss'}^a + \gamma \sum_{a'} \pi(s', a') Q^\pi(s', a') \right] \tag{7}
 \end{aligned}$$

The next logical step is to define the optimal value function and optimal action-value function. This starts with the assumption that there is an optimal policy π^* better than all of the others. One policy is better than another if its expected return is greater. The optimal value function V^* is thus defined as

$$V^*(s) = \max_{\pi} V^\pi(s) \tag{8}$$

for all $s \in S$. Similarly, the optimal action-value function is defined as

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a) \quad (9)$$

for all $s \in S$ and $a \in A(s)$. Q^* can be written in terms of V^* because Eq. (9) is the expected return of taking action a in state s and following an optimal policy for all subsequent steps; see Eq. (10):

$$Q^*(s, a) = E_{\pi}\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\} \quad (10)$$

The related Bellman optimality equations are listed in Eqs. (11) and (12):

$$V^*(s) = \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^*(s')] \quad (11)$$

$$Q^*(s, a) = \sum_{s'} \mathcal{P}_{ss'}^a \left[\mathcal{R}_{ss'}^a + \gamma \max_{a'} Q^*(s', a') \right] \quad (12)$$

Ideally, one would simply solve the set of linear Bellman optimality equations to acquire an optimal value function or optimal action-value function. However, that requires that the transition probabilities $\mathcal{P}_{ss'}^a$ and expected immediate rewards $\mathcal{R}_{ss'}^a$ be known. That is often not the case, unfortunately, and so other methods are often employed.

There are a number of ways to solve for the value or action-value functions. Three basic solution methods are dynamic programming (DP), Monte Carlo methods, and temporal-difference (TD) learning.

DP refers to algorithms that compute optimal policies given a perfect model of the environment. These are often computationally expensive and depend on a *perfect* model. DP algorithms include policy evaluation, policy improvement, policy iteration, value iteration, etc.

Monte Carlo methods estimate value functions and try to find optimal policies. One advantage of these methods over basic DP is that they require only experience and not perfect knowledge of the environment. Learning can be conducted online or in simulation with no prior knowledge of environment dynamics. These methods learn based on an episode-by-episode averaging of sample returns. Monte Carlo methods include Monte Carlo policy evaluation, Monte Carlo estimation of action values, Monte Carlo control (both on-policy and off-policy), etc.

TD learning can be thought of as a combination of ideas from both DP and Monte Carlo [19]. These methods learn from raw experience without the need for a model of the environment's dynamics. They bootstrap in the sense that estimates are updated *during* an episode based on other learned estimates. TD learning methods include TD prediction, Sarsa, *Q-learning*, actor-critic methods, etc.

The algorithm used in this paper is one step Q-learning, which is a common off-policy TD control algorithm. In its simplest form it is a modified version of

Eq. (13) and is defined by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (13)$$

where α is the learning rate and is $0 \leq \alpha \leq 1$. The Q-learning algorithm is illustrated as follows:

Q-Learning():

- Initialize $Q(s, a)$ arbitrarily
- Repeat (for each episode)
 - Initialize s
 - Repeat (for each step of the episode)
 - Choose a from s using policy derived from $Q(s, a)$ (e.g., ε -greedy policy)
 - Take action a , observe r, s'
 - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$
 - $s \leftarrow s'$
 - Until s is terminal
- Return $Q(s, a)$

The agent learns the greedy policy, defined as

$$\begin{aligned} &\varepsilon - \text{greedy policy} \\ &\text{if } (P > 1 - \varepsilon) \\ &\quad a = \arg \max_a Q(s, a) \\ &\text{else} \\ &\quad a = \text{rand}(a_i) \end{aligned} \quad (14)$$

where $0 \leq \varepsilon \leq 1$, and P is a uniformly distributed random number. As the learning episodes increase, the learned action-value function $Q(s, a)$ converges asymptotically to the optimal action-value function $Q^*(s, a)$. The method is an off-policy one as it evaluates the target policy (the greedy policy) while following another policy. The policy used in updating $Q(s, a)$ can be a random policy, with each action having the same probability of being selected. The other option is an ε -greedy policy, where ε is a small value. The action a with the maximum $Q(s, a)$ is selected with probability $1 - \varepsilon$; otherwise, a random action is selected.

If the number of the states and the actions of a reinforcement learning (RL) problem is a small value, its $Q(s, a)$ can be represented using a table, where the action value for each state-action pair is stored in one entry of the table. Because the RL problem for the morphing vehicle has states on continuous domains, it is impossible to enumerate the action value for each state-action pair. In essence, there are an infinite number of state-action pairs. One commonly used solution is to artificially quantize the states into discrete sets thereby reducing the number of state-action pairs the agent must visit and learn. The goal in doing this is to reduce the number of state-action pairs while maintaining the integrity of the learned action-value function. For a given problem, experimentation must be conducted to determine what kind of quantization

is appropriate for the states. In this chapter several increasingly larger quantizations are considered to determine what the largest allowable step size is. For the problem at hand, this becomes most important for keeping the number of state-action pairs manageable when more state variables are added to the thickness and camber in the form of other morphing parameters.

III. LEARNING ON A TWO- AND *N*-DIMENSIONAL CONTINUOUS DOMAIN

Q-learning on a continuous domain quickly becomes intractable when one considers that convergence of the algorithm to the optimal action-value function is only guaranteed if the agent visits every possible state an infinite number of times [14]. An agent would therefore visit an infinite number of states using an infinite number of actions an infinite number of times. Add in the fact that the states can be defined by an arbitrary finite number of continuous variables, and the so-called “curse of dimensionality” becomes a significant problem.

One way to cope with the inherent complexity of a continuous-domain learning problem is to discretize the state space by overlaying a pseudogrid. For the two-dimensional case, the state space is represented by Fig. 1. An arbitrary set of vertices $\{^1X, ^2X, \dots, ^jX, \dots\}$ are again introduced at uniform distances h_{x_1} or h_{x_2} apart. The actions available to the agent are again restricted as in the one-dimensional case. For the two-dimensional case, when the agent is at the

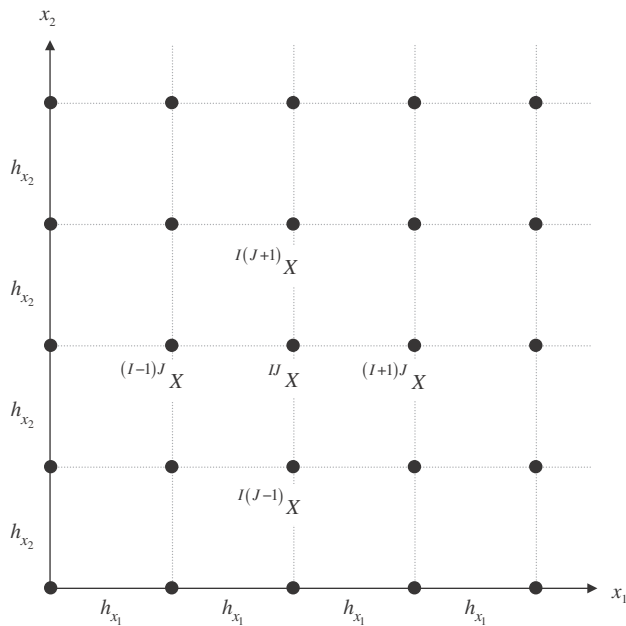


FIG. 1 Two-dimensional state space with overlaying pseudogrid.

If J th vertex $X = {}^JX$, it may only move to vertices ${}^{(I-1)J}X$, ${}^{(I+1)J}X$, ${}^{I(J-1)}X$, and ${}^{I(J+1)}X$, a total of four, or $2 * 2$, actions.

For this two-dimensional discrete case, let L_{x_1} and L_{x_2} denote the length in the x_1 and x_2 direction, respectively, of the continuous domain. This results in

$$\begin{aligned} N_{V_2} &= \left(\frac{L_{x_1}}{h_{x_1}} + 1 \right) \left(\frac{L_{x_2}}{h_{x_2}} + 1 \right) \\ &= \prod_{i=1}^2 \left(\frac{L_{x_i}}{h_{x_i}} + 1 \right) \end{aligned}$$

vertices. Therefore, there are

$$N_2 = 2 * 2 \prod_{i=1}^2 \left(\frac{L_{x_i}}{h_{x_i}} + 1 \right)$$

state-action pairs. This two-dimensional development is what will be used in the rest of this paper.

From here the formulation can be generalized to the N -dimensional case. For an N -dimensional continuous state space, an arbitrary set of vertices $\{{}^{11\dots 1}X, {}^{11\dots 2}X, \dots, {}^{NN\dots N}X\}$ is introduced at uniform distances $h_{x_1}, h_{x_2}, \dots, h_{x_N}$ apart. The actions are restricted to the two nearest vertices in any direction from the current vertex $X = {}^{JJ\dots J}X$, yielding a total of $2N$ actions available to the agent from any given vertex.

Now let $L_{x_1}, L_{x_2}, \dots, L_{x_N}$ denote the length in the x_1, x_2, \dots , and x_N directions, respectively. As a result, there are

$$N_{V_N} = \prod_{i=1}^N \left(\frac{L_{x_i}}{h_{x_i}} + 1 \right) \quad (15)$$

vertices. Therefore, there are

$$N_N = 2N \prod_{i=1}^N \left(\frac{L_{x_i}}{h_{x_i}} + 1 \right) \quad (16)$$

state-action pairs.

Discretizing the domain in this way can greatly simplify a learning problem. Intuitively, the larger h_{x_i} is, the fewer the number of vertices, resulting in fewer visits by the agent necessary to learn the policy correctly. Special care must be taken, however, in the choice of h_{x_i} and the definition of the goal the agent attempts to attain. If the only goal state lies between vertices, then the agent will be unable to learn the actions necessary to reach the goal state.

Discretizing the domain in this way can greatly simplify a learning problem, though limiting the action in this way is a significant assumption. In many applications, such as a robotics application, an action can change multiple state

variables. Limiting the actions simplifies the reinforcement learning problem so that focus during the early development and validation of this algorithm is on the algorithm itself rather than the complexity of the application. Future development of the algorithm, however, should eliminate this assumption.

IV. MULTIRESOLUTION STATE-SPACE DISCRETIZATION (AAG) FOR N DIMENSIONS

Discretizing a state-space for learning is beneficial in that it creates a finite number of state-action pairs the agent must visit. Generally, as the number of state-action pairs decreases, the rate of convergence increases [16]. However, fewer state-action pairs captures less detail of the environment. Also, using the method described in Sec. III limits the agent to the vertices. It is entirely possible that the goal the agent is seeking, or any other region of interest, does not lie on a vertex. This necessitates adding a range to the goal that encompasses one or more of the vertices in the state-space. These vertices within the goal range are pseudogoals (Fig. 2).

As the agent explores the coarsely discretized state space and garners rewards, it also notes the location of the pseudogoals. Once learning on the current discretization has converged, the region surrounding and encompassing the pseudogoals is rediscritized to a finer resolution such that $h_{x_{i_2}} < h_{x_{i_1}}$, where the subscript 1 denotes the initial discretization and subscript 2 denotes the second discretization. A new, smaller range is defined for the goal, and learning begins anew in the smaller state space. Figure 3 shows the rediscrization of the state space.

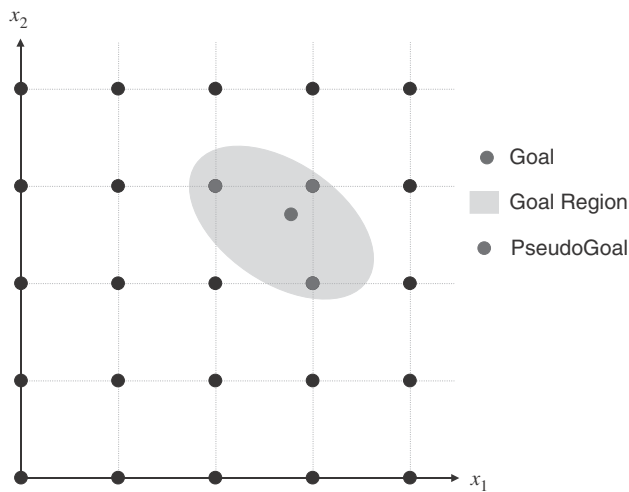


FIG. 2 Multiresolution state-space discretization — phase 1: coarse grid, large goal range.

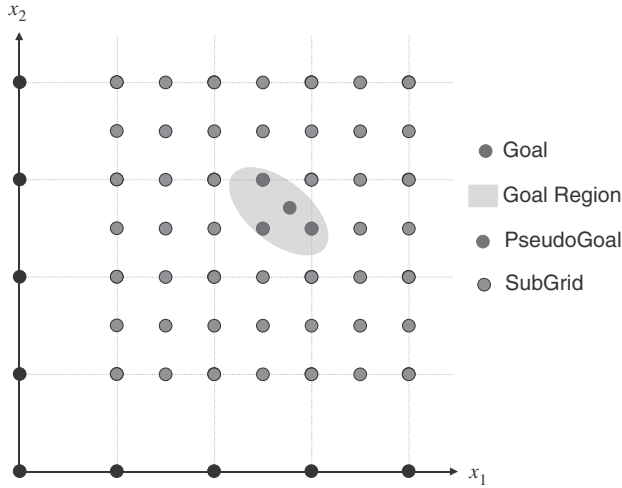


FIG. 3 Multiresolution state-space discretization — phase 2: finer grid, smaller goal range.

This method can then be generalized for the N -dimensional case. Let $L_{x_1}^j, L_{x_2}^j, \dots, L_{x_N}^j$ denote the length in the x_1, x_2, \dots , and x_N directions, respectively, and the superscript j denotes the resolution of the discretization in which 1 is the coarsest and M is the finest. The vertices for each resolution are then set at distances $h_{x_1}^j, h_{x_2}^j, \dots, h_{x_N}^j$ apart. These terms effectively define the fineness of resolution level j . Equations (15) and (16) can then be modified to calculate the number of vertices and state-action pairs for this method, as shown in Eqs. (17) and (18).

When the multiresolution learning is complete, there are

$$N_{V_N} = \sum_{j=1}^M \left[\prod_{i=1}^N \left(\frac{L_{x_i}^j}{h_{x_i}^j} + 1 \right) \right] - \sum_{j=1}^{M-1} \left[\prod_{i=1}^N \left(\frac{L_{x_i}^{j+1}}{h_{x_i}^j} + 1 \right) \right] \quad (17)$$

vertices. Therefore, there are

$$N_N = 2N \left\{ \sum_{j=1}^M \left[\prod_{i=1}^N \left(\frac{L_{x_i}^j}{h_{x_i}^j} + 1 \right) \right] - \sum_{j=1}^{M-1} \left[\prod_{i=1}^N \left(\frac{L_{x_i}^{j+1}}{h_{x_i}^j} + 1 \right) \right] \right\} \quad (18)$$

state-action pairs. Notice the second term of each equation excises the duplicate vertices from one level of discretization to the next. Also, note that if the full state space were simply discretized by the finest level of $h_{x_i}^M$, there would be

$$N_{V_{N_{\text{fine}}}} = \prod_{i=1}^N \left(\frac{L_{x_i}^1}{h_{x_i}^M} + 1 \right) \quad (19)$$

vertices and

$$N_{N_{\text{fine}}} = 2N \left[\prod_{i=1}^N \left(\frac{L_{x_i}^1}{h_{x_i}^M} + 1 \right) \right] \quad (20)$$

state-action pairs. It can be shown that $N_{V_N} < N_{V_{N_{\text{fine}}}}$ and $N_N < N_{N_{\text{fine}}}$ by a significant amount, the magnitude of which is determined by the factor by which each subsequent discretization is reduced from the earlier one.

It is known that the time to convergence for Q-learning increases exponentially as the complexity of the problem, that is, state-action pairs, increases. This method reduces a learning problem to a series of smaller learning problems with relatively few state-action pairs, on the order of several orders of magnitude less. Rather than one large problem that will take a great deal of time to converge, there are several quickly converging smaller problems.

V. MULTIPLE GOAL REGIONS

In many problems it is entirely possible that more than one distinct region of the state space contains pseudogoals. When discretizing the state-space from level j to $j + 1$ using the multiresolution discretization method just described, one option is simply to have the algorithm define $L_{x_1}^{j+1}, L_{x_2}^{j+1}, \dots, L_{x_N}^{j+1}$ such that the region encompasses *all* of the pseudogoals. The drawback of this approach is that if there are distinct groupings of pseudogoals, then this approach discretizes to finer degree regions that contain nothing of interest to the agent. Continuing to learn in these regions is essentially wasted effort.

Another approach is to separate and discretize each grouping of pseudogoals individually. Using the earlier notation, this approach would essentially yield $L_{x_1}^{j_1+1}, L_{x_2}^{j_1+1}, \dots, L_{x_N}^{j_1+1}$ for one group of pseudogoals as denoted by the j_1 , $L_{x_1}^{j_2+1}, L_{x_2}^{j_2+1}, \dots, L_{x_N}^{j_2+1}$ for the second group, and $L_{x_1}^{j_n+1}, L_{x_2}^{j_n+1}, \dots, L_{x_N}^{j_n+1}$ for the n th distinct group of pseudogoals.

The set of pseudogoals recorded by the agent is separated into its groups using a general clustering technique from the statistical analysis toolbox in MATLABTM. Let R denote the recorded pseudogoals to be clustered. R is an $m \times n$ matrix with m observations of n variables or m pseudogoals defined by n state variables in the context of machine learning. The first step in the clustering process is to determine the pairwise distance of all of the observations using the following equation for the Euclidean distance:

$$d_{rt}^2 = (\mathbf{x}_r - \mathbf{x}_t)(\mathbf{x}_r - \mathbf{x}_t)' \quad (21)$$

When arranged into a row vector \mathbf{y} , this yields a vector length of $m(m - 1)/2$ arranged in the order $[(2, 1), (3, 1), \dots, (n, 1), (3, 2), \dots, (n, 2), \dots, (n, n - 1)]$.

A hierarchical cluster tree Z , where Z is an $(m - 1) \times 3$ matrix, is then created from the Euclidean distances encoded in \mathbf{y} . Cluster indices linked in pairs are

denoted in columns 1 and 2 of Z to form a binary tree. Leaf nodes, or the singleton clusters from which all higher clusters are built, are numbered from 1 to m and are the original observations. Column 3 contains the linkage distance, as calculated by Eq. (22), between two merged clusters. When a new cluster is formed, a row is added to Z and is assigned the index $m + I$, where $Z(I, 1 : 2)$ contains the clusters indices that form cluster $m + I$ and $Z(I, 3)$ contains the linkage distance between the two.

$$d(r, t) = \min[(\mathbf{x}_{r_i} - \mathbf{x}_{t_j})(\mathbf{x}_{r_i} - \mathbf{x}_{t_j})] \quad [i \in (1, \dots, n_r), j \in (1, \dots, n_t)] \quad (22)$$

Clusters can then be formed and assigned cluster numbers from these linkages by comparing the length of each link in Z with the average length at the same level in the hierarchy.

Each cluster represents a distinct set of pseudogoals contained in R . Using these clusters, the multiresolution discretization method can now be applied to each individual cluster in turn, allowing the agent to learn each region in finer detail and neglect the uninteresting regions in between.

VI. POLICY COMPARISON

The multiresolution discretization method provides a means of learning the action-value function $Q^\pi(s, a)$ for a fixed policy π in progressively finer detail. Rather than blindly allowing the agent to learn for the entire number of user-defined episodes, stopping criteria based on the learned policy are introduced.

In Q-learning all of the information is stored in the form of the action-value matrix, often in the form of a table. The learned greedy policy itself is not represented explicitly or with any sort of model. However, the policy and associated value function can be easily extracted from the action-value function in a few simple steps. There exists a simple relationship between the action-value function and the greedy policy, namely,

$$\pi(s) = \arg \max_a Q(s, a) \quad (23)$$

where $\pi(s)$ is the action associated with the maximum preference over the set of actions for the state.

In addition, a representation of the value function can then be easily calculated:

$$V(s) = \max_a Q(s, a) \quad (24)$$

for the tabular action-value function. This relationship will be used for visual analysis in later sections.

Two stopping criteria are added to the Q-learning algorithm. These two criteria are a direct policy comparison and a performance-based policy comparison, which are periodically applied to the learned action-value function to determine if the action-value function has converged to a usable data set. Both criteria use the

relationships in Eq. (23) for the tabular action-value function. Both of the stopping criteria introduced in this section must be met for learning at the current level of discretization to be terminated; otherwise, learning continues. These criteria are described in the following sections.

A. DIRECT POLICY COMPARISON

The direct policy comparison stopping criteria is a simple and expedient way to track the change in the policy extracted from an action-value function as that function evolves during learning. This policy comparison is carried out in a short series of steps:

1. Pause learning after n episodes.
2. Extract current greedy policy $\pi_i(s)$ from action-value function, where i is the number of elapsed episodes divided by n .
3. Directly compare $\pi_i(s)$ and $\pi_{i-1}(s)$.
4. If change in policy $\Delta\pi$ is $< \varepsilon$, then stopping criteria #1 is achieved.

Here ε is a small number and is set as 5% for the examples in this chapter, and n is a user-defined number of episodes on the order of 100. This same series of steps holds for the approximated action-value function. Also, after learning is initialized and the first set of n episodes elapse, the policy $\pi_1(s)$ is extracted and stored only as there is no $\pi_0(s)$. This comparison method is fully utilized starting after the second set of n elapsed episodes. $\Delta\pi$ is determined by comparing the most recently stored policy with the penultimate stored policy and calculating the percentage of change from one to the other.

It is entirely possible that a lack of change in the extracted policy is not an indication that the action-value function has converged to a usable function. It could simply be a momentary aberration and would begin to change again if another n episode is allowed. To prevent this from occurring, a second stopping criterion is introduced.

B. PERFORMANCE-BASED POLICY COMPARISON — MONTE CARLO SIMULATION

The second stopping criterion implemented is based on the performance of the current policy. During the pause in learning after every n episode, the policy comparison is conducted and then this performance-based policy comparison. The performance-based policy comparison measures performance by conducting a set of Monte Carlo simulations. It is referred to as Monte Carlo in the sense that initial conditions are taken from a uniform distribution, and a large number of simulations are conducted and recorded. In each simulation the agent is initialized in a random nongoal state within the region of the current level of discretization. It then uses the current learned greedy policy, meaning it exploits its current knowledge of the state space to navigate through the state

space to find the goal. A *success* occurs when the agent navigates from the random initial state to a goal state without encountering a boundary. A *failure* occurs when the agent either encounters the outermost boundary of the state space, the boundary of the current level of discretization, or gets “lost” and wanders around the state space. This simulation is conducted a predefined number of times, usually 500 simulations, and each success is recorded. The success percentage is then calculated using Eq. (25):

$$\% \text{ Success} = \frac{\# \text{ of Successes}}{\text{Total } \# \text{ of Simulations}} \quad (25)$$

When this success percentage is above some threshold, usually 98%, the second stopping criterion is satisfied. Both the first and the second stopping criterion must be met for the learning at the current level of discretization to be terminated and learning continued at a finer level of discretization as necessary.

VII. BENCHMARK DYNAMIC SYSTEM EXAMPLE — INVERTED PENDULUM

The inverted pendulum is a classic system on which to test new control algorithms. It is a simple unstable system that requires a certain amount of finesse to control. The simple inverted pendulum system shown in Fig. 4 has four degrees of freedom: cart position x , cart velocity \dot{x} , pole angular position θ , and pole angular velocity $\dot{\theta}$. This system has one equilibrium point, which is unstable at $\theta = 0$ deg.

As just stated, this simple system, and derivatives thereof, is often used to test new controllers. For example, the system in which two pendulums are to be balanced was addressed using a double cerebellar model articulation controller (CMAC) network [21]. The authors effectively learned to balance the system by first splitting it up into two single pendulum problems and then combining them with some extra learning to balance the double system. There are also many who have controlled this system with fuzzy logic controllers in one form or another. One such reference used the Takagi–Sugeno model to do the full swing-up and balance maneuver of the

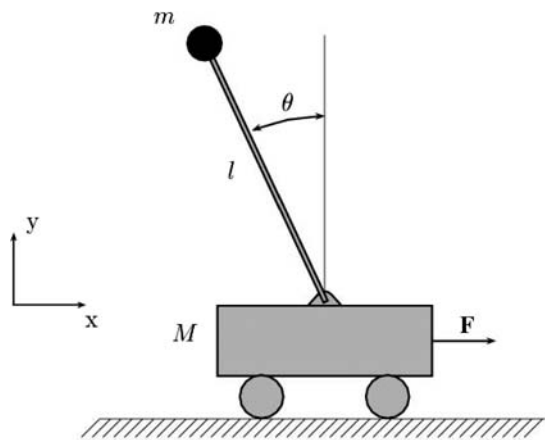


FIG. 4 Inverted pendulum system.

single inverted pendulum with friction [22]. Another used a fuzzy controller to do the swing-up maneuver and the balancing after a certain threshold using a linear quadratic regulator.

Q-learning has been used a number of times as well. A hierarchical reinforcement learning controller for the double inverted pendulum was designed by Zheng et al., which uses a similar setup as the CMAC method, but with standard Q-learning as the algorithm of choice. The system is again split into two single pendulum problems and combined into the double pendulum [23].

There are many more references regarding control via linear quadratic regulators, fault-tolerant adaptive control, all manner of reinforcement learning methods, modern control, and classical control. In general, this type of system is a benchmark control problem on which controllers ought to be tested. It is versatile in the sense that it can be made more difficult by using the full nonlinear equations of motion and/or by adding friction or damping effects. It can be made simpler by linearizing about the equilibrium point and/or negating damping effects.

A. INVERTED PENDULUM MODEL

The inverted pendulum is simulated as a simple, nonlinear dynamic system. Friction is assumed negligible. The derived equations of motion for this system are

$$\begin{aligned}\ddot{x} &= \frac{-mg \cos \theta \sin \theta - m \dot{\theta}^2 l \sin \theta + F}{M + m - m \cos^2 \theta} \\ \ddot{\theta} &= \frac{F - (M + m)g \tan \theta + m \dot{\theta}^2 l \sin \theta}{-\left(\frac{Ml + ml}{\cos \theta}\right) + ml \cos \theta}\end{aligned}\quad (26)$$

where m is the mass of the ball, M is the mass of the cart, l is the length of the rod, θ is the angle between the rod and vertical, $\dot{\theta}$ is the angular rate of the rod, $\ddot{\theta}$ is angular acceleration of the rod, x is the horizontal distance of the cart, \ddot{x} is the horizontal acceleration of the cart, g is acceleration due to gravity, and F is the horizontal force applied to the cart. The constants for the system are listed in Table 1.

The Q-learning controller determines the sign and magnitude of the force to be applied to the system in intervals of $\Delta t = 0.05$ s. In essence the controllers act as sampled data regulators as they attempt to balance the pole around the equilibrium point $\theta = 0$ deg.

B. INVERTED PENDULUM CAST AS A REINFORCEMENT LEARNING PROBLEM

Casting a dynamic system as a reinforcement learning problem takes careful consideration. The degrees of freedom chosen to be the state variables, such that $s \in S$, to be used by the learning algorithm must adequately capture the dynamics of the system and the goal to be achieved. The dynamic system, and subsequently

TABLE 1 CONSTANTS OF THE INVERTED PENDULUM SYSTEM

Parameter	Value
M , kg	5
m , kg	1
l , m	5
g , m/s^2	9.81

the state variables, constitute the *environment* with which the agent interacts. This system has four possible degrees of freedom, the interdependence of which are captured in the two equations of motion. If one were to rush into casting the system as a reinforcement learning problem, one would include all degrees of freedom as state variables $(x, \dot{x}, \theta, \dot{\theta}) \in S$. This would be necessary if the goal were to balance the pole around $\theta = 0$ deg and restrict the cart to a finite track. Because this is not case for this formulation of the problem and only balancing the pole around $\theta = 0$ deg is of interest with no restrictions on the cart, only the angle and angular velocity of the pendulum mass are considered to be state variables $(\theta, \dot{\theta}) \in S$. Note that both equations of motion must be retained and simulated as they are interdependent and the state variables of interest appear in both equations.

The only input into the system is a horizontal force applied to the cart. Thus, the *action* available to the agent is the force to be applied to the system, that is, $(+F, -F) \in A$. The forces are set at the user’s discretion. The action space can be restricted to two actions only or have as many as ten or more. The main effect of action selection is on the complexity of the problem because the more actions available to the agent, the more trials necessary to learn which actions are appropriate for the given state.

The *reward* r received by the agent from the environment is based on the current state of the system. Because this is a simple system, the reward structure is kept simple (see Table 2). The reward is set up such that if the agent manages to propel the pendulum near the goal, it receives a positive reward. Additional positive rewards are received if the pendulum is maintained near the goal for multiple

TABLE 2 INVERTED PENDULUM Q-LEARNING EXAMPLE REWARD
STRUCTURE

Bounds	Reward
$ \theta \leq 2$ deg	1
2 deg $< \theta \leq 12$ deg	0
$ \theta > 12$ deg	−1

time steps. This encourages the agent to learn to balance the pendulum for as many time steps as possible. A negative reward is received if the pendulum falls beyond some angle, and no reward or a neutral reward is received if the angle of the pendulum lies between the two bounds.

For this problem the state space is *quantized* rather than discretized. Quantized means that the dynamic system is simulated in continuous time, and at pre-defined intervals the state is noted, and the update to the action-value function determined by the learning algorithm is applied to the nearest learning storage state in the action-value function.

C. NUMERICAL RESULTS

The agent is given the task of learning to balance an unstable system for as many time steps as possible. By setting up the system as a reinforcement learning problem just described, the agent is set to learn how to control the system. Additional parameters for the learning problem are listed in Table 3. The agent is allowed 5000 episodes over which to learn the controller for the current goal. The initial state for each episode is random and within the neutral or positive bounds of the state space. Each episode is simulated for a total of 50 s, and every 0.05 s the state is noted, and the action-value function updated. If the pendulum falls beyond the negative bounds, the episode is terminated and a new episode begun.

For this example, the agent learns using two algorithms. The agent first uses Q-learning to establish a baseline controller. In the second part, the agent uses Q-learning with AAG in an attempt to learn a better controller. The two controllers are evaluated in a number of ways. First, it is helpful to consider a value function approximated from the learned action-value function. The value function can show how the learning differs when learning parameters or discretization is modified. The policy can be extracted from the action-value function as described in Sec. VI using Eq. (23). This equation represents the action associated with the maximum preference or action value for each state. An approximation of the

TABLE 3 INVERTED PENDULUM Q-LEARNING PARAMETERS

Parameter	Value
Episodes	5000
Δt	0.05 s
t_f	50 s
α	0.01
γ	0.7

**TABLE 4 INVERTED PENDULUM Q-LEARNING
SIMULATION INITIAL CONDITIONS**

State	Value
x	0 m
\dot{x}	0 m/s
θ	0.5 deg
$\dot{\theta}$	0 deg/s

value function based on both the action-value function and the extracted policy can then be found with Eq. (24) from Sec. VI.

Because this is a dynamic system, it is also informative to simulate the system with the learned controller in action. Both controllers are evaluated for simulations of 10 and 300 s. The initial conditions for the simulations are listed in Table 4. Control is applied in the same way that it is learned. Every $\Delta t = 0.05$ s an action or force is selected based on the highest preference in the action-value function given the current state; see Eq. (23). That force is applied for 0.05 s, and then a new action is selected. The 10-s simulation is intended to show an up-close view of the workings of the controller, whereas the 300-s simulation is intended to prove that controller can balance the pendulum for an extended period of time, in this case for 6000 time steps.

1. CASE 1: Q-LEARNING

The controller learned using the Q-learning algorithm is restricted to only a couple possible actions. These action are the following forces

$$(-F, F) \in A \tag{27}$$

where $F = 15N$ for these examples. The reward structure for this case is that listed in Table 2. The states were quantized to every 1 deg and 1 deg/s for angular position and velocity, respectively.

The value function and policy for this case are shown in Fig. 5. The actions for the policy are color coded and are listed in Table 5. The value function shows a marked peak crest in the goal region where the agent received positive rewards. This encourages the agent to attempt to remain in this area. There is a sharp drop-off in value beyond the goal region. There are also negative regions near the outer bounds of the state space. The visual representation of the policy shows that the agent probably needs more learning time. One would expect more continuous regions of a given action. The figure and intuition of the system suggests that for a fully converged policy, the upper right half of the state space should be light grey and the lower left half dark grey. Because of the dynamic nature of the system, however, there is no coordinated way to have the agent

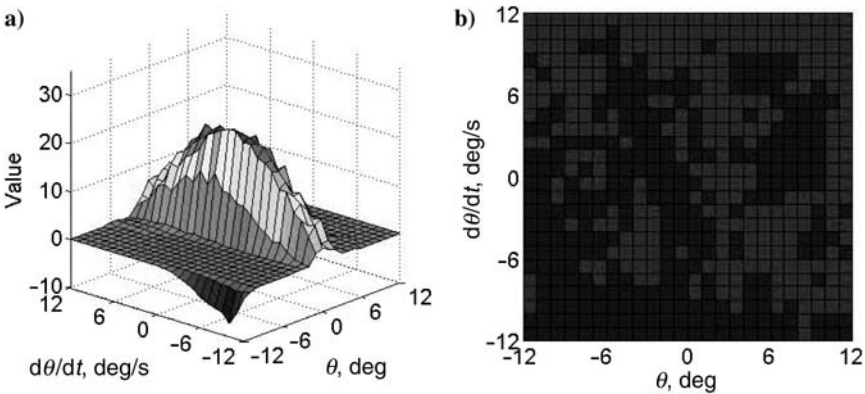


FIG. 5 Basic inverted pendulum controller: a) value function and b) policy representation.

focus on these regions aside from initializing the many episodes in these regions. For more complex problems, there is no guarantee that there would be large regions in which one particular action is ideal. Therefore, artificially forcing the agent to initialize in these regions for this simple problem is counterproductive because the method is unlikely to scale up to a more complex problem.

Either way, the true test for this kind of learned controller is to actually test the controller. Figures 6 and 7 show the time history and phase diagram for the 10- and 300-s simulations, respectively. The time history in Fig. 6 illustrates the jagged nature of this controller resulting from the restricted number of control inputs. Despite this, the learned controller balances the pendulum within the goal region of $|\theta| \leq 2$ deg as evidenced by both the time history and the phase diagram. The angular velocity does spike close to 8 deg/s about 9 s into the simulation, but the controller is able to recover the system and maintain balance.

The time history in Fig. 7 shows a similar response as the controller continues to balance the pendulum. The pendulum is balanced for the full 300 s and appears to be stable, but does not get much closer than the oscillation between about $\theta = -1$ and 2 deg with the angular velocity approaching 8 deg/s many times. The agent achieved its goal of learning a controller that balances around $\theta = 0$ deg and within $|\theta| \leq 2$ deg, but the response leaves much room for improvement.

TABLE 5 INVERTED PENDULUM POLICY COLOR SCHEME

Action	Color
$-F$	Dark grey
F	Light grey

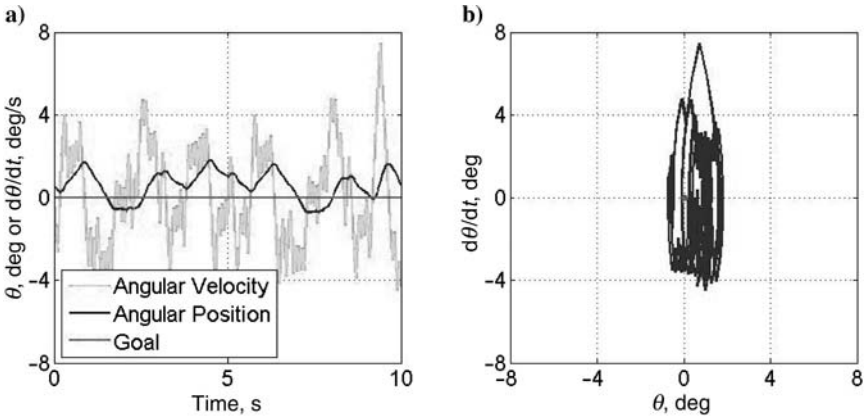


FIG. 6 Basic inverted pendulum controller, 10 s: a) time history and b) phase diagram.

2. CASE 2: Q-LEARNING WITH AAG

The controller learned using Q-learning with AAG is a little more complicated. There are three levels of discretization or quantization on which the agent is allowed 5000 episodes for learning in each. The action sets and state quantizations for each level are listed in Table 6.

Initial attempts at integrating AAG into this problem had the goal regions dependent solely on θ as is used in case 1. Simulations using the learned controller were not satisfactory, and so a more complex scheme was implemented. This scheme is shown in Fig. 8. For this case the goal regions are dependent on both

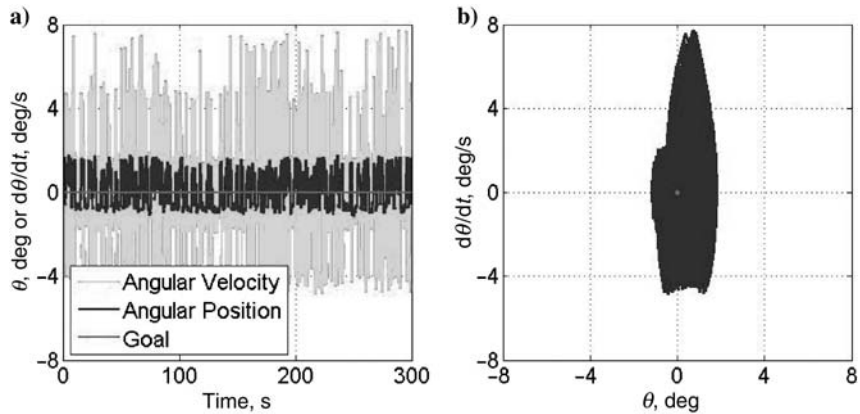


FIG. 7 Basic inverted pendulum controller, 300 s: a) time history and b) phase diagram.

TABLE 6 INVERTED PENDULUM ACTION SETS AND STATE QUANTIZATIONS

Discretization Level	Action Set	State Quantization ($\theta, \dot{\theta}$)
1	$(-15N, 15N) \in A$	(1 deg, 1 deg/s)
2	$(-5N, 5N) \in A$	(0.2 deg, 0.2 deg/s)
3	$(-1N, 1N) \in A$	(0.04 deg, 0.04 deg/s)

θ and $\dot{\theta}$. The equations that describe the bounds shown in Fig. 8 are listed in Table 7. Figure 8 also shows all of the quantized states where information was stored by the agent. When learning, these bounds also act as the outer bounds at finer levels of discretization. For example, the first level of quantization uses the outer bounds listed in Table 2 and the goal bounds listed in Table 7. For the second level of quantization, the linear equations describing the goal region for the first level of quantization now act as the outer bounds for the second level, and the goal region for the second level of quantization is in effect.

The value function and policy for this case are shown in Fig. 9. The actions for the policy are the same as the preceding case and are listed in Table 5. The value function for this case shows a more extreme peak at $(\theta, \dot{\theta}) = (0, 0)$ and a similar crest in the goal region. The levels of quantization are also evident in the value function in the blocky regions on the outer edges of the crest slope and beyond and the fine details near the center of the crest and the peak. The visual representation of the policy again shows that the agent probably needs more learning time. However, the regions of a given action are more contiguous than the preceding case. The patterning of the policy suggests that good control of the pendulum when exploiting learned knowledge is sensitive to initial conditions.

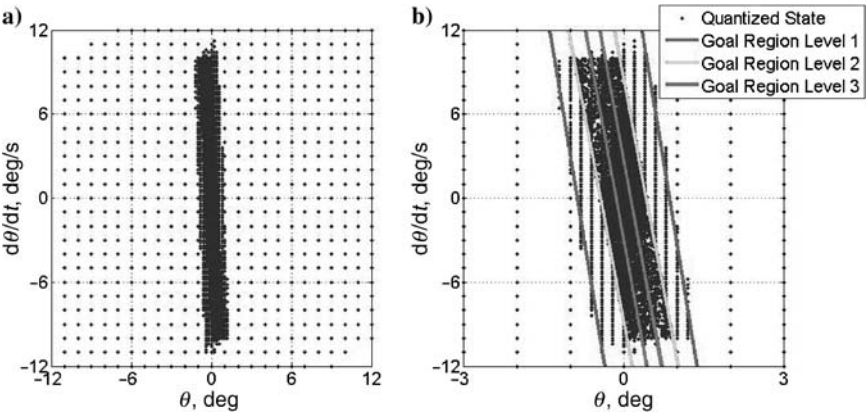


FIG. 8 AAG inverted pendulum controller: a) quantized states and b) goal regions.

TABLE 7 INVERTED PENDULUM AAG GOAL REGION EQUATIONS

Discretization Level	Lower-Bound Equation	Upper-Bound Equation
1	$\dot{\theta} = -23.3\theta - 20$	$\dot{\theta} = -23.3\theta + 20$
2	$\dot{\theta} = -20\theta - 8.6$	$\dot{\theta} = -20\theta + 8.6$
3	$\dot{\theta} = -20.7\theta - 2.9$	$\dot{\theta} = -20.7\theta + 2.9$

This controller is also tested for a short time period and an extended time period as shown in Figs. 10 and 11, respectively. The controller is implemented such that the actions available to the controller are dependent on the state, as they were during learning. This means that if the state of the pendulum is outside the level 1 goal region bounds, then the agent may only use the level 1 actions. When the state of the pendulum is within the level 1 or level 2 goal region bounds, the controller may use the level 2 or level 3 actions, respectively. If the state is within the level 3 goal region bounds, then the controller may use the level 3 actions. The time history in Fig. 10 shows that this controller is able to balance the pendulum much closer to the equilibrium point than the previous controller. The agent balances to within an angle of ± 0.3 deg within 5 s of the simulation. The angular velocity shows clearly the effects of the controller. In the preceding case the angular velocity time history was very jagged as the controller sought to balance the pendulum by alternating between two large opposing forces. The angular velocity shown in Fig. 10 shows evidence of the controller using the smaller forces, which results in a smoother time history and better control of the pendulum. The angular velocity does not get as large as in the previous case. The largest spike in angular velocity is 3.3 deg/s and quickly reduces to

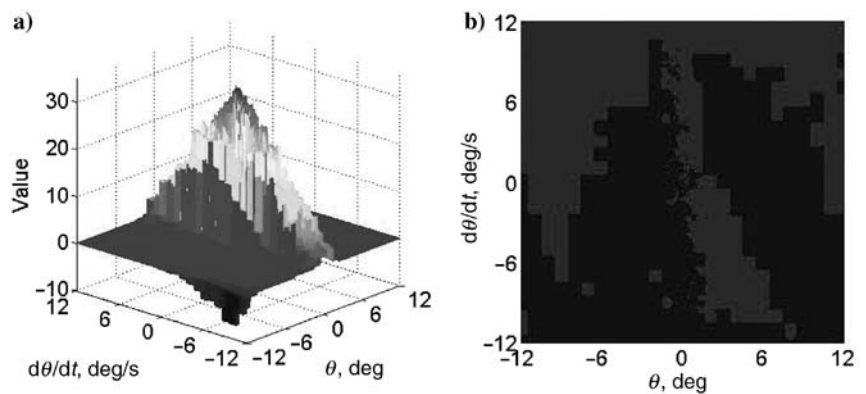


FIG. 9 AAG inverted pendulum controller: a) value function and b) policy representation.

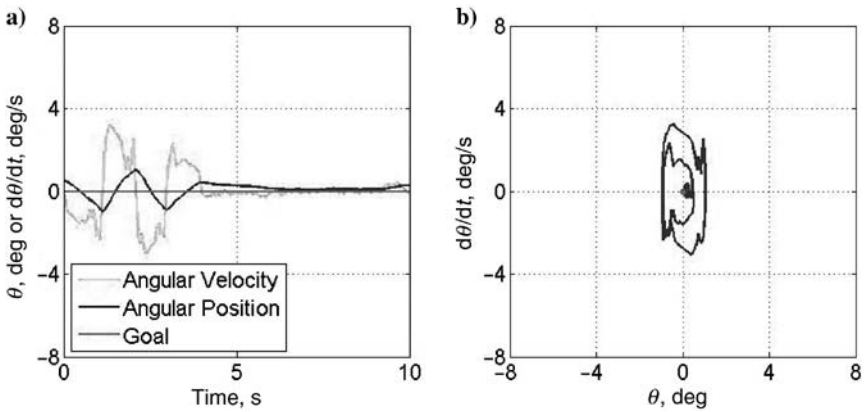


FIG. 10 AAG inverted pendulum controller, 10 s: a) time history and b) phase diagram.

within ± 0.5 deg/s. All of these details are also shown in the phase diagram. The small dot at (0,0) indicates the ideal goal of a perfectly balanced pendulum. The phase diagram shows that the pendulum oscillates briefly before good balance is achieved near this point.

The time history in Fig. 11 shows that the same trend continues as time elapses. The controller maintains balance of the pendulum to within an angle of ± 0.3 deg for the duration of the 300 s. The angular velocity also remains low, within ± 0.5 deg/s, after the initial balancing for the duration of the simulation. On closer inspection of the time history and phase diagram, it is evident that the pendulum is near the equilibrium, but tends to unbalance to a positive

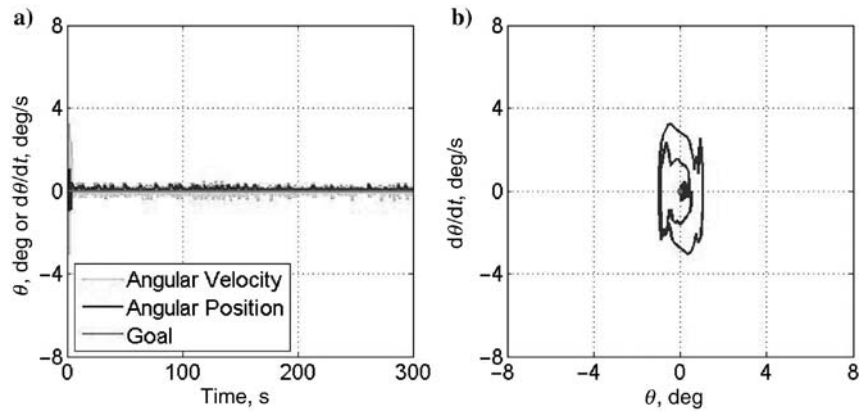


FIG. 11 AAG inverted pendulum controller, 300 s: a) time history and b) phase diagram.

angle, though it quickly recovers. This trend is due to the velocity of the cart, which is not considered in this formulation. The cart velocity stabilizes around $x = 5$ m/s, which in turn causes the pendulum to drift to a positive angle. The current controller is able to balance the pendulum despite this fact, but it could be avoided by putting constraints on x and \dot{x} and including them in the learning problem, thereby creating a more complex problem.

VIII. AUTONOMOUS SOARING UPDRAFT/THERMAL EXAMPLE

Large birds and glider pilots extend flight duration, increase cross-country speed, improve range, and conserve energy by exploiting thermals caused by convection in the lower atmosphere. The problem of learning an updraft field by an autonomous soaring UAV is cast as a reinforcement learning problem tasked with learning the commands that produce the actions to move from some position to the middle of an updraft, based on the vertical velocity of the surrounding air. The unique aspect of this paper is the modification of AAG to allow the multiresolution learning around *multiple* regions of interest.

Section VIII.A discusses the thermal model to be cast as the environment of the reinforcement learning problem. Then the full thermal location reinforcement learning problem is described, followed by a series of examples that exercise the many aspects of the problem. Results for learning an updraft field with a single thermal in two dimensions, multiple thermals in two dimensions, and multiple thermals in three dimensions are all presented.

A. THERMAL MODEL

The environment is an updraft model developed by NASA Dryden Flight Research Center for use in developing autonomous soaring UAVs [24]. Autonomous soaring refers to a UAV using updrafts to extend flight duration, increase speed, improve range, or conserve energy. The model was created using data collected at the National Oceanic and Atmospheric Administration Surface Radiation Station. The model is a statistical representation of the convective velocity scale w^* and the convective mixing-layer thickness z_b , which were used to determine updraft size, vertical velocity profile, spacing, and maximum height [24]. The full development of the model can be found in [24].

B. THERMAL LOCATION CAST AS A REINFORCEMENT LEARNING PROBLEM

The updraft model is cast as a reinforcement learning problem by defining the model itself as the environment. The model uses Cartesian coordinate inputs x and y (and z if desired) supplied by the reinforcement learning agent as its current state, to calculate the distance to each of the randomly spaced updraft

centers w_{c_i} , using simple Euclidean distance:

$$d_i = \sqrt{(w_{c_{ix}} - x)^2 + (w_{c_{iy}} - y)^2} \quad (28)$$

The minimum distance is extracted, and the vertical velocity at the current location is calculated and passed to the agent. Figure 12 shows a representative updraft field. It is assumed that the agent is able to move throughout this updraft field and store learned information of the location and path to updrafts.

The updraft model and the reinforcement learning agent interact significantly during both the learning stage, when the location and path to updrafts are learned, and the operational stage, when the agent is asked to move through the updraft field to the nearest updraft by transferring from state to state. The reinforcement learning agent must learn the series of actions necessary to command both the x and y positions of the agent to the nearest updraft. The two parts of the system interact as follows.

The agent interacts with its environment by choosing actions from a set of admissible actions. The state space is discretized in the manner described in Sec. III, and so these actions include incremental changes in the position of the agent. Thus, the agent is effectively restricted to movement between adjacent vertices. An example of admissible action in this context is the following. The agent chooses to move in the x_1 direction from vertex ${}^I X$ in the two-dimensional problem. For the initial discretization, the two possible actions in the x_1 direction are defined as follows:

$$\begin{aligned} A_{11}^1 &\equiv {}^{(I+1)}X - {}^I X = h_{x_1}^1 \\ A_{12}^1 &\equiv {}^{(I-1)}X - {}^I X = -h_{x_1}^1 \end{aligned} \quad (29)$$

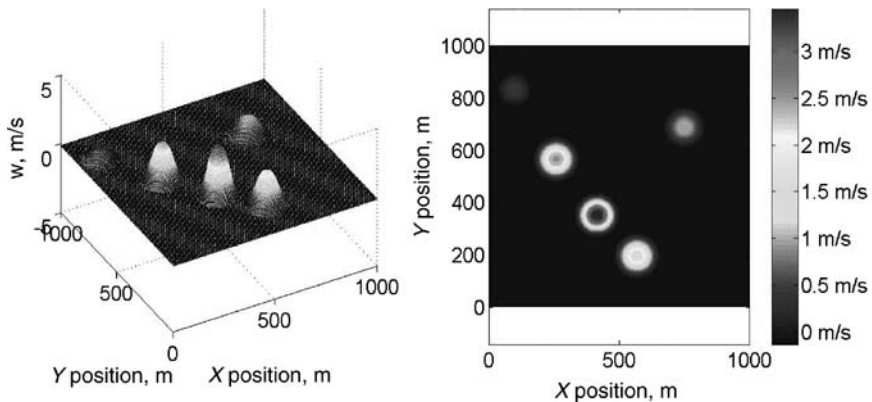


FIG. 12 Representative updraft field.

Equation (29) can be summarized by saying the initial admissible actions in the x_1 direction are $A_1 = \pm h_{x_1}^1$. Similar relationships can be found for the x_2 direction. Admissible actions in the other direction is $A_2 = \pm h_{x_2}^1$. For this problem, the state variables are the Cartesian coordinates, and the actions are incremental changes in these coordinates. These state variables are listed in Table 8.

It is noted that learning in this manner, moving from vertex to adjacent vertex, produces an action-value matrix specifying preferences for local transitions that could be used to create a path by specifying a series of actions from some arbitrary state to a goal state. Specifying the actions in this way allows for the agent to learn to avoid hazards in the updraft field, such as down-drafts (negative vertical velocity of the air) or ground obstacles (should they be included in the model). If actions were specified by transitions from one specific position or state to another, for example, from $(x, y)_1 = (0 \text{ m}, 100 \text{ m})$ to $(x, y)_2 = (800 \text{ m}, 750 \text{ m})$ and thus hoping to encounter an updraft at point 2, the agent would then have an initial state and a final state, but would need some form of path planning algorithm to chart a path from point 1 to point 2. The problem with doing so in this reinforcement learning problem is that the agent does not have a priori knowledge of where updrafts and hazards are located between the two points, which would make planning a path to avoid hazards or utilize updrafts unnecessarily difficult.

The goal w_g of the agent for this problem is defined by the vertical velocity of the thermals. The agent is essentially tasked with finding the center of any and all thermals in the updraft field. Position changes of the agent in the updraft field necessitate evaluation of the local vertical velocity of the surrounding air. Changes in vertical velocity and proximity to updrafts located during learning define the reward, which leads to the reward being based on two equations depending on where in the state space the agent is located. Because of the nature of the updraft field, there is a uniform velocity, or sink velocity, outside of the actual updrafts as indicated by the dark field in Fig. 12. Therefore a gradient-based reward function dependent on vertical velocity is not feasible. In this region, the reward is instead based on proximity to the updrafts. The multiresolution discretization method records the states encountered by the agent that are categorized as “goal” states. Once at least one goal state is

TABLE 8 THERMAL LOCATION AXIS DEFINITIONS

x_i	Definition
x_1	X position, m
x_2	Y position, m
x_3	Z position, m

TABLE 9 INITIAL UPDRAFT FIELD LIMITS

Initial limit	Lower	Upper
X position	0	1000
Y position	0	1000
Z position	100	1000

found, the reward function in the sink region is

$$r_u = \frac{0.001}{g_f^m} \left(\sqrt{\sum_{i=1}^n (s_{g_i} - s_i)^2} - \sqrt{\sum_{i=1}^n (s_{g_i} - s'_i)^2} \right) \quad (30)$$

where r_u is the reward in the sink velocity region, s_g is the nearest goal state to the current state, s, s' is the next state according to the Q-learning algorithm, g_f^m is the discretization factor to the power of the current level of fineness m , and 0.001 is a scaling factor. Prior to finding the first goal state, rewards in this sink region are simply 0. As the learning progresses and more goal states are found, it is likely that the s_g used for a given state will change. The action-value function will update accordingly and will still converge in this region.

When the agent encounters and maneuvers within an updraft, the reward function is based on a vertical velocity gradient defined by the following equation:

$$r_w = |w_g - w_s| - |w_g - w'_s| \quad (31)$$

where r_w is the reward in the updraft, w_g is the goal vertical velocity, w_s is the vertical velocity in the current state, and w'_s is the vertical velocity in the next state determined by the Q-learning algorithm. The region of interest is the goal (of which there can be multiple) of $w = 4$ m/s, and the associated initial range is ± 4 m/s. The initial boundary limits of the state space are listed in Table 9.

C. NUMERICAL RESULTS

The purpose of the numerical example is to demonstrate the learning performance of the reinforcement learning agent utilizing the multiresolution state-space discretization method and the policy-based stopping criteria with multiple goal regions. The agent is to learn and update the action-value function as the discretization becomes increasingly finer in the various regions of interest. The updraft field to be learned is that represented by Fig. 13. The location of the updrafts and their individual strengths are randomly generated by the updraft model. The goal, as determined from the updraft field, is $w_g = 4$ m/s, and the initial range is $g_{r_1} = 4$ m/s. The final range for the goal in this problem is $g_{r_3} = 1$ m/s.

The agent follows a 100% exploration policy for this example. The agent is to learn the action-value function with multiple levels of discretization. Each finer

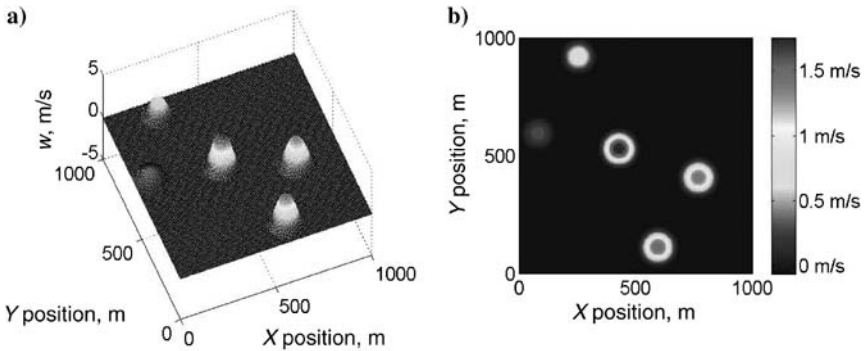


FIG. 13 Updraft field for multiple thermals in two-dimensions case 2.

discretization is determined using a preset factor g_f applied to the coarser discretization, such that $h_{x_i}^{j+1} = g_f h_{x_i}^j$. The number of levels of discretization or resolution as defined earlier is M . The parameters for this problem are listed in Table 10.

The learning is paused, and PC is applied every 200 episodes. Analysis of the learning is conducted in the following manner: the dimensionality of the multire-solution action-value function is compared with that of the full state-space discretized at the finest level, the Monte Carlo simulation performance results are analyzed, and the final value function and policy are considered.

1. DIMENSIONALITY

The dimensionality for this problem is analyzed in two ways. The first is visually considering the distribution of states in the state space visited by the agent. The

TABLE 10 THERMAL LEARNING PARAMETERS FOR THERMALS IN TWO-DIMENSIONS

Parameter	Value
Episodes	5000
α	0.01
γ	0.7
g_f	0.2
w_g	0.5
M	3
$(h_{x_1}^1, h_{x_2}^1)$	(50 m, 50 m)

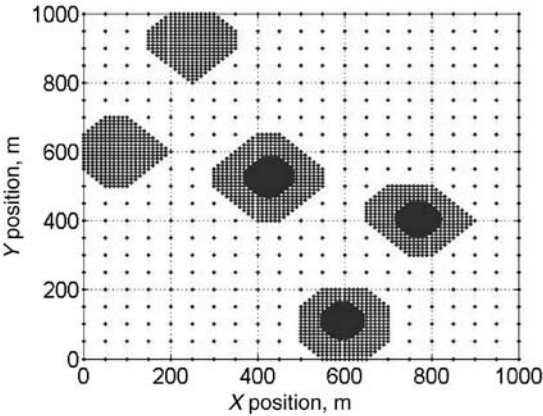


FIG. 14 Multiple thermals in two-dimensions with AAG and PC: visited state in the updraft field.

second is considering the number of states and state-action pairs. Figure 14 shows the distribution of visited states for this problem. There are more states in and around each thermal as a result of the multiresolution method

with the densest packing of states around the center of each thermal. Notice that the two thermals located at $(x, y) = (80, 590)$ and $(x, y) = (255, 920)$ do not have regions with states from the third level of discretization. The reason is that these two thermals are fairly weak (maximum velocities < 1 m/s) compared to the strongest thermal, so that at the higher levels of discretization with the more restricted goal regions focusing on stronger updraft velocities there are no goal states and thus no region of interest to explore at the next level of discretization.

The numerical dimensionality is listed in Table 11. The multiresolution method reduces the number of state-action pairs the agent must visit by two orders of magnitude. AAG allows the agent to focus most of its attention on these five smaller regions of interest individually rather than wasting time in areas of the state space between these regions.

2. MONTE CARLO SIMULATION

Figure 15 shows the results of Monte Carlo simulation analysis for this problem. Each level of discretization is allowed a possible 5000 episodes, which means that the agent is allowed 5000 episodes for each region of interest at each level of discretization. The thermal locations and their corresponding designation in Fig. 15

TABLE 11 MULTIPLE THERMALS IN TWO-DIMENSIONS WITH AAG AND PC: STATES AND STATE-ACTION PAIRS

Resolution	States	State-Action Pairs
Multi	7,845	31,380
Single	251,001	1,004,004

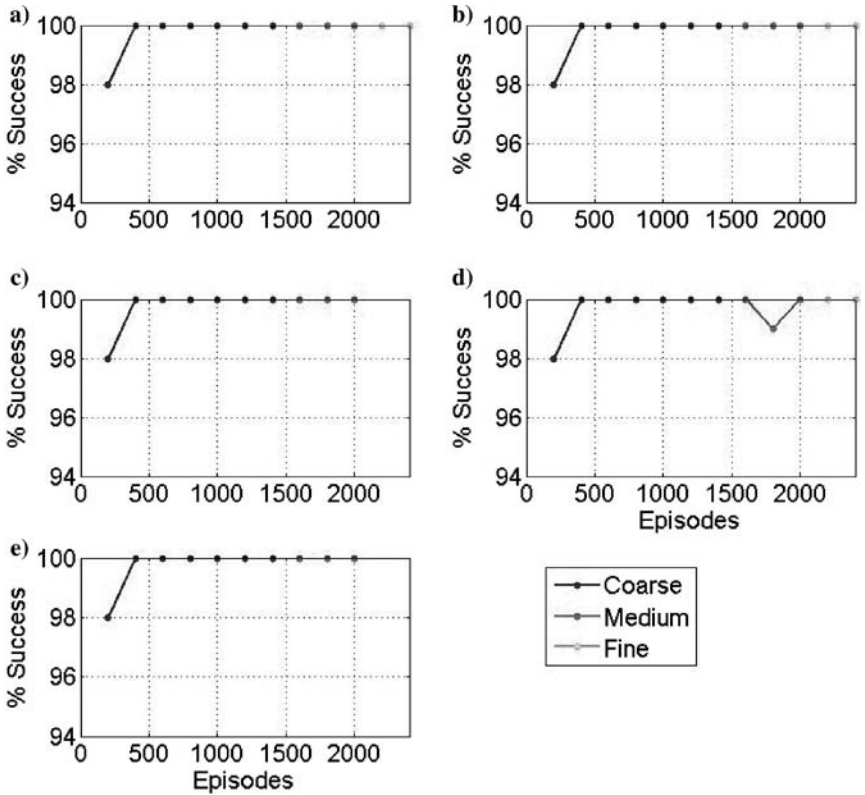


FIG. 15 Multiple thermals in two-dimensions with AAG and PC: Monte Carlo simulation results.

TABLE 12 MULTIPLE THERMALS IN TWO-DIMENSIONS WITH AAG AND PC: THERMAL LOCATIONS

Designation	X Position, m	Y Position, m
a	430	530
b	770	405
c	80	595
d	595	110
e	255	920

TABLE 13 MULTIPLE THERMALS IN TWO DIMENSIONS WITH AAG AND PC: NUMBER OF EPISODES USED

Discretization	Regions Explored	Episodes per Region	Total Episodes per Level	Total Episodes allowed per Level
Level 1	1	1400	1400	5,000
Level 2	5	600	3000	25,000
Level 3	3	400	1200	15,000
Total episodes	—	—	5600	90,000

are listed in Table 12. Each region of interest is tested separately for the second and third discretization. PC registers a success for the initial discretization if the agent moves from its initial state to *any* region of interest.

The figures show that learning on the first level of discretization converges quickly, within 400 episodes. The first level is not terminated, however, until after 1400 episodes. The reasoning for this delay is that the policy is still changing by at least 5% every 200 episodes. It is not until the 1400th episode that the policy has converged and has a performance of 100% success. The policy converges even more quickly for the second level of discretization for each thermal. The policy for the area surrounding each region of interest reaches 100% success almost immediately, and learning terminates in 600 episodes. Only three of the thermals—a, b, and d—needed further discretization and learning based on the final goal range of $g_{r_3} = 1$ m/s. The figures show that the policy again converges almost immediately, in 200 episodes, and learning is terminated after 400 episodes. Table 13

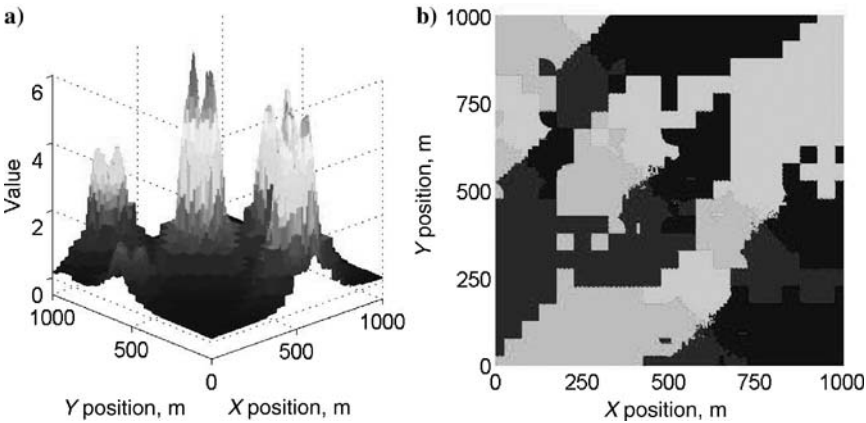


FIG. 16 Multiple thermals in two-dimensions with AAG and PC: a) value function and b) policy representation.

TABLE 14 THERMAL POLICY COLOR SCHEME

Action	Color
$-h_{x_1}$	Blue
$+h_{x_1}$	Cyan
$-h_{x_2}$	Yellow
$+h_{x_2}$	Red

summarizes how the total possible number of episodes is determined. According to Table 13, only 5600 were needed of the total 90,000 episodes possible, which is a 93.7% reduction.

3. VALUE FUNCTION AND POLICY ANALYSIS

In both images in Fig. 16, the various levels of discretization are evident in the blocky nature of the surfaces. The policy representation is color coded by action, defined in Table 14. The large squares in the policy representation and the large block structures in the value function representation indicate areas where the largest discretization was applied. The areas with finer discretization, especially around the bottom-left region, are evident in the more refined surface gradation and color separation in the value function and policy, respectively. As in the preceding case, each crest in the value function mirrors the strength of the thermal in that location. With the additional episodes at the finer levels of discretization, there are no obvious areas that would cause the agent to get stuck and be unable to proceed to the nearest thermal.

IX. CONCLUSIONS

The results show that the learning for the multiresolution method reduces the dimensionality of the problem by two orders of magnitude and allows the agent to focus on the regions of interest. The Monte Carlo simulation showed that the multiresolution method reaches a 98% success rate or greater within 400 episodes for the coarsest level of discretization, within 200 episodes for the second level of discretization, and within 200 episodes for the third level of discretization. The policy comparison reduced the total number of episodes by 93.7% from the user-defined number of episodes per level of discretization.

Conceptually, a problem with the full state space discretized at the finest level would take many thousands or tens of thousands of episodes more to reach this level of convergence. This method is successful in learning fine detail in multiple distinct goal regions and greatly reducing the time for convergence, increasing the rate of convergence, and achieving a goal with the small range of ± 1 m/s. This

method essentially reduced the larger problem by almost 95%, making it a much more tractable learning problem with a series of smaller problems.

ACKNOWLEDGMENTS

This work was sponsored (in part) by the Air Force Office of Scientific Research, USAF, under Grant/Contract number FA9550-08-1-0038. The technical monitor is Fariba Fahroo. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

REFERENCES

- [1] Bentivegna, D. C., Atkeson, C. G., and Cheng, G., "Learning to Select Primitives and Generate Sub-Goals from Practice," *Proceedings of the 2003 IEEE/IRIS International Conference on Intelligent Robots and Systems*, Vol. 1, 2003, pp. 946–953.
- [2] Simsek, O., Wolfe, A. P., and Barto, A. G., "Identifying Useful Subgoals in Reinforcement Learning by Local Graph Partitioning," *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 816–823.
- [3] Clausen, C., and Wechsler, H., "Quad-Q-Learning," *IEEE Transactions on Neural Networks*, Vol. 11, No. 2, Feb. 2000, pp. 279–294.
- [4] Dietterich, T. G., "The MAXQ Method for Hierarchical Reinforcement Learning," *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998, pp. 118–126.
- [5] Dietterich, T. G., "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition," *Journal of Artificial Intelligence Research*, Vol. 13, 2000, pp. 227–303.
- [6] Mehta, N., Soumya, R., Tadepalli, P., and Dietterich, T., "Automatic Discovery and Transfer of MAXQ Hierarchies," *Proceedings of the 25th International Conference of Machine Learning*, 2008, pp. 648–655.
- [7] Diuk, C., Strehl, A. L., and Littman, M. L., "A Hierarchical Approach to Efficient Reinforcement Learning in Deterministic Domains," *Proceedings of the Fifth International Joint Conference on Autonomous Agent and Multiagent Systems*, 2006, pp. 313–319.
- [8] Elfving, S., Uchibe, E., Doya, K., and Christensen, H. I., "Evolutionary Development of Hierarchical Learning Structures," *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 2, April 2007, pp. 249–264.
- [9] Makar, R., Mahadevan, S., and Ghavamzadeh, M., "Hierarchical Multi-Agent Reinforcement Learning," *Proceedings of the Fifth International Conference on Autonomous Agents*, ACM Press, 2001, pp. 246–253.
- [10] Kirchner, F., "Q-Learning of Complex Behaviours on a Six-Legged Walking Machine," *Robotics and Autonomous Systems*, Vol. 25, No. 4, Nov. 1998, pp. 253–262.
- [11] Chen, C., Li, H., and Dong, D., "Hybrid Control for Robot Navigation: A Hierarchical Q-Learning Algorithm," *IEEE Robotics & Automation Magazine*, Vol. 15, No. 2, 2008, pp. 37–47.

- [12] Ghavamzadeh, M., and Mahadevan, S., "Learning to Communicate and Act Using Hierarchical Reinforcement Learning," *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004*, 2004, pp. 1114–1121.
- [13] Vigorito, C. M., and Barto, A. G., "Hierarchical Representations of Behavior for Efficient Creative Search," *AAAI Spring Symposium: Creative Intelligent Systems 2008*, 2008, pp. 135–141.
- [14] Watkins, C. J. C. H., and Dayan, P., "Learning from Delayed Rewards," Ph.D. Dissertation, Univ. of Cambridge, Cambridge, England, UK, 1989.
- [15] Lampton, A., Niksch, A., and Valasek, J., "Multi-Resolution State-Space Discretization Method for Q-Learning Applied to Highly Reconfigurable Systems," *Journal of Machine Learning Research*, (in review).
- [16] Lampton, A., Niksch, A., and Valasek, J., "Reinforcement Learning of Morphing Airfoils with Aerodynamic and Structural Effects," *Journal of Aerospace Computing, Information, and Communication*, Vol. 6, No. 1, Jan. 2009, pp. 30–50.
- [17] Lampton, A., Niksch, A., and Valasek, J., "Reinforcement Learning of a Morphing Airfoil — Policy and Discrete Learning Analysis," *Journal of Aerospace Computing, Information, and Communication*, Vol. 7, No. 8, Aug. 2010, pp. 241–260.
- [18] Lampton, A., Niksch, A., and Valasek, J., "Morphing Airfoil with Reinforcement Learning of Four Shape Changing Parameters," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA, Reston, VA; also AIAA Paper 2008-7282, Aug. 2008.
- [19] Sutton, R., and Barto, A., *Reinforcement Learning — An Introduction*, MIT Press, Cambridge, MA, 1998, pp. 9, 51–80, 89–100, 111, 133, 148–150.
- [20] Stark, H., and Woods, J. W., *Probability and Random Processes with Applications to Signal Processing*, Prentice–Hall, Upper Saddle River, NJ, 2002, pp. 421–423.
- [21] Zheng, Y., Luo, S., and Lv, Z., "Control Double Inverted Pendulum by Reinforcement Learning with Double CMAC Network," *Proceedings of the 18th International Conference on Pattern Recognition*, 2006, pp. 639–642.
- [22] Tao, C., Taur, J. S., Hsieh, T. W., and Tsai, C. L., "Design of a Fuzzy Controller with Fuzzy Swing-up and Parallel Distributed Pole Assignment Schemes for an Inverted Pendulum and Cart System," *IEEE Transactions on Control Systems Technology*, Vol. 16, No. 6, Nov. 2008, pp. 1277–1288.
- [23] Zheng, Y., Luo, S., Lv, Z., and Wu, L., "Control Double Inverted Pendulum by Hierarchical Reinforcement Learning," *Proceedings of the ISPC*, 2004, pp. 1614–1617.
- [24] Allen, M. J., "Updraft Model for Development of Autonomous Soaring Uninhabited Air Vehicles," AIAA Paper 2006-1510, Jan. 2006.

Motion Planning Under Uncertainty

Ali-Akbar Agha-Mohammadi*

Texas A & M University, College Station, Texas 77843

Sandip Kumar[†]

MathWorks, Natick, Massachusetts 07160

Suman Chakravorty[‡]

Texas A & M University, College Station, Texas 77843

I. INTRODUCTION

A basic problem that has to be faced and solved by autonomous vehicles, on which this chapter will focus, is the problem of *motion planning*. It involves generation and execution of a plan for moving around an environment towards a designated goal, or to accomplish a desired task, while avoiding collisions with obstacles in the environment. Moreover, it is desirable to optimally use the available resources to achieve the goal, thereby minimizing some “cost” function.

There are many established techniques to solve motion planning problem in a deterministic framework ranging from optimal control method [1], grid world approaches [2] to randomized sampling-based motion planners [2–4]. However, real-world systems are not deterministic, and their evolution involves uncertainty due to surrounding environments or internal parameter variance. Hence, in the real world, these deterministic algorithms are applied along with some trajectory tracking techniques, which accounts for the uncertainty in the system [5].

Another approach to solving the motion-planning problem is to take uncertainty into account while solving the problem. Introduction of uncertainty in the motion-planning problem increases the complexity of the problem. Uncertainty in the system can be caused by two scenarios, one due to sensing uncertainty and the other due to process uncertainty. Sensing uncertainty arises from sensor noise during measurements or an uncertain environment, that is, partial knowledge of obstacle locations in the given environment. The process uncertainty is the motion uncertainty due to presence of stochastic forcing in the system dynamics and controls.

*Graduate Student, Department of Computer Science and Engineering.

[†]Application Support Engineer, 3 Apple Hill Drive.

[‡]Associate Professor, Department of Aerospace Engineering.

Dynamical systems with uncertainty evolves as a stochastic process that has the Markovian property, that is, the evolution is memoryless, and the future and past states do not influence the system evolution given the current state. Solving an optimization problem for such a Markovian stochastic process can be modelled in a mathematical framework termed *Markov decision process* (MDP) [6]. Researchers have attempted to solve MDPs using *dynamic programming* (DP) [7, 8] and *reinforcement learning* (RL) techniques [9]. The DP methods are model-based methods whereas RL is a model-free fashion of approaching the same problem.

Introduction of sensing uncertainty in the system transforms the MDP into a so-called *partially observed Markov decision process* (POMDP) [7, 8]. In the absence of perfect state information, an estimator is needed to provide an estimate of system state, based on which the decision making is performed. Usually, the estimator provides a probability distribution over all possible system states. This distribution is called *belief*, and the space of all possible beliefs is called *belief space*, denoted by \mathbb{B} . It can be shown that the POMDP can be stated as an MDP in the belief space [10, 11]. Thus, also in the absence of perfect state information the solution of planning problem is obtained by solving a DP equation. However, this DP equation is defined over the belief space.

A. TECHNICAL APPROACH

The motion planning problem is a sequential decision-making problem, and optimal control is the most general framework for solving such a problem. We wish to solve the motion-planning problem involving uncertainty in the form of process uncertainty and stationary stochastic maps. The motion-planning problem can be formulated as a MDP, if the uncertainties in the robot motion and environments can be modeled probabilistically. However, MDPs are virtually intractable for anything but small to moderate state spaces as they suffer from “curse of dimensionality” [12]. It means the complexity of solving MDPs grows exponentially as the dimensionality of the problem increases, and hence it is nearly impossible to solve even without constraints. Because of the introduction of constraints (which in robotic motion-planning problem framework means obstacles, velocity and acceleration rates, torque and force saturation, and limited domain), there is no structured technique to accomplish the planning. Hierarchical methods [13, 14] have tried to resolve the issue of dimensionality by introducing “distinguished states” (called “nodes” here) and invoking options/policies at these states that can only terminate at these states. Using this transformation, the original large MDP can be transformed into a semi Markov decision process (SMDP), which needs to be solved only at these states, and, hence, the computational burden is reduced drastically on the stochastic optimization algorithm.

In parallel, deterministic robotic motion-planning problems have been solved using randomized sampling-based algorithms [2], like probabilistic roadmaps (PRM) [3] and rapidly exploring random trees (RRT) [4]. The essential idea

behind these techniques is to sample configurations of the robot in the free configuration space \mathcal{C}_{free} and attempt to connect them using local planners (like straight-line planners) and hence generate a topological graph $\mathcal{G}(V, E)$ with vertices V being the sampled configurations and edges E of the graph being the connecting path between vertices developed using the local planners. Given the graph, the solution to the motion-planning problem involves discrete search on the developed graph (as in PRM) or biasing the graph towards the goal configuration (as in RRT). They have proved extremely powerful in solving high dimensional problems that were previously unsolvable using traditional deterministic planning techniques. Unfortunately, these techniques were designed for a deterministic framework and are not robust to systems with uncertainty. Various attempts have been made to incorporate stochastic maps, process uncertainty, and sensing uncertainty in the problem. This research is discussed in the related work section, further in this section.

In this chapter we present a generic framework to be used on suitable spaces to solve motion-planning problems in the presence of uncertainty and then apply onto problems related to motion/process uncertainty or/and sensing uncertainty.

First, we solve the motion-planning problems in presence of motion/ process uncertainty and stochastic maps. We generalize the PRM method to sequential decision making problems with process uncertainty and stochastic maps such that the benefits of PRM may be realized for the robotic motion-planning problem under process and map uncertainty, named generalized probabilistic roadmaps (GPRM), a sampling-based feedback motion planner. The solution methodology will involve design with scales for continuous state space and continuous control spaces and will handle constraints in the state space of the robot. Our technique will utilize feedback solutions to the lowest-level local planner of the hierarchical SMDP-based solution technique. The use of feedback controllers in the presence of uncertainty ensures that the trajectories converge to the goal location, thus mitigating the effects of process uncertainty. Our technique will decompose the problem into a two-layer hierarchical SMDP. The “lowest level” will consist of feedback solutions between node states, and the “top level” consists of a topological graph, with nodes as vertices, on which solution is searched. (We will be using stochastic DP [7, 8].) Feedback planners of *lowest level* provide information (e.g., transition probabilities and costs) to the *top level*, and with probabilities involved they induce an MDP at the *top level* on these node states, resulting in an SMDP, the solution to which can be found using *stochastic DP*. Solution to this stochastic DP results in a hybrid feedback solution to the motion planning problem in the presence of process uncertainty and stochastic maps. Finally, we would also like to provide the formal analysis of the proposed algorithm in terms of probabilistic completeness, as this is a sampling-based motion planner.

Second, in this chapter, we provide a solution methodology to solve the motion-planning problem in the presence of motion/process as well as sensing uncertainty. We generalize the PRM framework to the feedback controller-based information-state roadmap (FIRM) that takes into account both motion and sensing

uncertainties. The probability density function (pdf) over the state is called *belief* or *information-state*. The FIRM is constructed as a roadmap in belief space, where its nodes are small subsets of belief space and the edges of FIRM are Markov chains in belief space. It is the first method that generalizes the PRM to belief space in such a way that the incurred cost on different edges of roadmap are independent of each other, while providing a straightforward approach to sample reachable nodes in belief space. These properties are a direct consequence of utilizing feedback controllers in the construction of FIRM. Planning under motion and sensing uncertainty is essentially a POMDP. An important contribution of FIRM is that it breaks the curse of history in such POMDPs and provides the optimal policy over the roadmap instead of only a nominal path.

Optimal substructure: From the algorithm point of view, the independence of cost of different edges on the graph paraphrases the optimal substructure property: The optimal substructure property holds for a problem, if the optimal solution of the problem can be constructed by the combination of optimal solutions of its subproblems [15].

Optimal substructure assumption: To return an optimal solution of a problem using dynamic programming methods, the problem has to satisfy the optimal substructure property. Moreover, in the successive approximation schemes that solve the DP for the shortest path problem, such as Dijkstras algorithm, the optimal substructure assumption has to hold [16]. In other words, the underlying assumption in Dijkstras algorithm to find the shortest path is that the cost of any subpath has to be independent of what precedes it and what succeeds it. As mentioned, in direct transformation of sampling-based methods to the case with uncertainty, this assumption breaks.

One of the main characteristics (and contributions) of the sampling-based feedback motion planning algorithms we present here is that they preserve the optimal substructure property. In other words, we aim in generalizing sampling based in deterministic setting to sampling methods in stochastic setting such that the optimal substructure assumption still holds. Doing so, we can use DP or its successive approximation schemes to find the optimal solution for the desired MDP problem.

B. RELATED WORK

Motion planning of robots while avoiding obstacles in the workspace has been an active area of research for the last several decades in the robotics and artificial intelligence community. Classical motion planning can roughly be divided into the following three different deterministic approaches [17]: 1) cell decomposition, 2) roadmaps, and 3) potential field methods. In potential field methods a collision-free trajectory is generated by the robot according to “forces” defined as the negative gradient of a potential function. The cell decomposition and roadmap techniques are deterministic methods because the environment of the robot is sampled or discretized in a deterministic manner. However, the problems are

PSPACE-hard [2], and to circumvent this computational complexity, randomized sampling-based methods known as PRMs were introduced [3, 18]. PRM techniques usually do not take the dynamics of the robotic platform into account, and this case can lead to serious performance issues. To address these issues, RRT was introduced as a randomized sampling-based planner that takes into account the dynamics of the mobile robot [2, 12] while building a tree of dynamically feasible trajectories in the free space of the robot.

The randomized PRM and RRT techniques have resulted in the solution of motion-planning problems in very high-dimensional state spaces, which were hitherto unsolvable using deterministic motion-planning techniques. However, both PRM and RRT are open-loop planners designed for perfectly known robot models/workspaces, and our primary motivation in this work is to generalize these two techniques to generate feedback motion planners that are robust to uncertainties in the robot motion model and the map.

Uncertainty in robotic systems usually can stem from three sources: 1) motion uncertainty, which is also called process or dynamic uncertainty and resulted from the noise that affects system dynamics; 2) sensing uncertainty, which is also referred to as imperfect state information and can be caused by the noise that affects the measurements made by sensors; 3) uncertainty in environment map, such as uncertain obstacle locations or uncertain location of features (information sources) in the environment. Furthermore, the robot motion-planning problem can be formulated as an MDP if the uncertainties in the robot model and the environment are modeled probabilistically. However, MDPs are virtually intractable for anything but small to moderate state spaces because they are subject to the famous “curse of dimensionality” [12]. In particular, it is nearly impossible to solve these problems in continuous state and control spaces even without constraints. In the presence of constraints, there are no structured techniques for accomplishing the planning. One approach to resolving the issue of dimensionality is through the use of hierarchical methods, an approach that is seen in most biological systems. A variety of methods for solving large MDPs in a hierarchical model-free manner has been developed, and the field of research is known as hierarchical reinforcement learning (RL) [13, 14].

These methods, instead of taking actions, invoke policies or options at each state, which continue until termination. Moreover, if it is assumed that these temporally abstract policies can terminate only at one of a few “distinguished states,” then the original large MDP can be transformed into a significantly smaller SMDP that needs to be solved only at the distinguished states and thus drastically reduces the computational burden of the dynamic programming algorithms used to solve the problem. However, following issues are key in the formulation and solution of an SMDP: 1) how the node states are chosen, 2) how the local options are designed, and 3) how the generalized cost and transition probabilities of the options are estimated. The model-free techniques estimate the control without estimating the SMDP parameters through simulation or online training. However, questions 1 and 2 are not addressed in these techniques. In this work

we will model the motion-planning problem as an SMDP and will answer the questions just posed.

Further in this section we would like to discuss work closely related to solving the motion-planning problem in the presence of stochastic maps, process uncertainty, and sensing uncertainty.

1. RELATED WORK: STOCHASTIC/UNCERTAIN MAPS

In this section, we review research related to map uncertainty.

In [19], the need to plan on uncertain maps is addressed. They propose an *uncertainty roadmap*, where they maintain an upper and lower bound on the map probabilities and refine them incrementally as needed.

In [20], a sampling-based motion planner was proposed with sensing uncertainty built into the planning process. This is an utility guided planner, and they refined the uncertain map model using sensing actions.

In [21], an algorithm is proposed to compute motion plans that are robust to uncertain maps, which is an extension of PRM. The map uncertainty is evaluated using a feature-based extended kalman filter (EKF) algorithm. Monte Carlo simulations are done for the open-loop local planner to detect collision and use A^* search over the roadmap. The results shown in the work show high failure rate for a three-degrees-of-freedom (DOF) mobile robot with uncertain maps.

In [22], a *particle RRT* (pRRT) algorithm that can deal with uncertainty in the model is proposed. The RRT is extended using particle-based techniques, that is, each extension is simulated multiple times under various likely conditions of the environment. The likelihood of a path involves simulating particles with uncertainty in domain. The work especially dealt with uncertainty in parameters used to define the workspace.

2. RELATED WORK: PROCESS UNCERTAINTY IN DYNAMICAL MODELS

In this section, we consider research that has tried to account for process uncertainty in the planning.

In [23], a medial-axis-based PRM is proposed, which generates trajectories that are robust to modeling errors because samples on the medial axis of the plane maximize their distance from obstacles; however, they do not explicitly consider uncertainty in the PRM.

In [24], a variant of PRM is proposed called the *belief roadmap* (BRM), which solves a POMDP and uses a Kalman-filter-based estimator. This work does not account for process uncertainty.

In [25], a motion-planning algorithm to deal with process uncertainty for nonholonomic dynamical systems is proposed. Obstacle location uncertainty is solved using a minimum clearance approach. It poses the overall problem as an MDP by discretizing the space as a grid and solve it using DP.

In [26], a *stochastic motion roadmap* is proposed to deal with process uncertainty. As variants of PRM, they sample the configurations and consider discrete actions at each state, and transition probabilities are calculated. Using these transition probabilities, they search the roadmap using DP and also carry the notion of a probability of success. Hence, they do not consider stochastic maps, and the control space is discrete.

In [27], a planning algorithm is proposed to account for uncertainty in the dynamics of the vehicle. They propose a hierarchical planning approach, characterize noise as a function of controller type, terrain type and control input, introduce error dynamics, assume line follower, and search on the discretized free space using A* based-ARA* algorithm. The notion of motion uncertainty while following a trajectory in a corridor is calculated using the distance from the obstacles.

In [28], the problem of planning paths guaranteed to be safe in the presence of boundedness in process and sensing uncertainty is addressed. They propose an RRT-based algorithm, *set-RRT*, which uses a set of configurations. They attempt to solve the sensing uncertainty problem and propose to use proprioceptive sensors instead of exteroceptive sensors and rely on prediction to ensure collision avoidance. Not considering exteroceptive sensors leads to unbounded growth of belief uncertainty thereby leading to severe performance degradation, which has been noted in [24].

In [29], a technique is proposed to account for uncertainties in the motion primitives used by a maneuver automaton. The framework of a maneuver automaton and a dynamic programming formulation was extended to explicitly account for uncertainty in each of the motion primitives. The motion primitives considered were trim conditions and maneuver, and uncertainty was considered in trim parameters and maneuver displacement and duration.

In [30], uncertainty was considered in sensing, localization, and mapping in the motion-planning problem. They propose to use RRT along with simulated particle-based SLAM algorithm to solve this problem. They estimate the collision likelihood using the particle-filter-based framework.

Reference [31] introduced the notion of node-based approach to solve motion-planning problem under uncertainty. They introduced nodes as a subset of robot's configuration space where position sensing and motion control are perfect, outside which the sensing is null and control is imperfect. They propose node design and an approach to solve the problem using geometrical analysis. They used grid-based motion planners.

3. RELATED WORK: PROCESS AND SENSING UNCERTAINTY IN DYNAMICAL MODELS

The framework we propose for planning under motion and sensing uncertainty is called FIRM, and here in this section we briefly discuss the planning approaches that are related to the FIRM. The class of methods that are most related to FIRM considers both motion and sensing uncertainties in planning, where the environment (location of obstacles and information sources) is

known. Censi et al. [32] propose a planning algorithm based on graph search and constraint propagation on a grid-based representation of the space. Platt et al. in [33] plan in continuous space by finding the best nominal path through nonlinear optimization methods. In the linear quadratic Gaussian motion-planning (LQG-MP) method of Van den Berg et al. [34], the best path is found among the finite number of RRT paths by simulating the performance of LQG on all of them. Authors extend the framework also to the roadmaps in [35]. Moreover, Prentice and Roy [36] and Huynh and Roy [37] are also roadmap-based methods based on PRM approach, where the best path is found through breadth-first search on the Belief roadmap. However, on all of these roadmap-based methods the “optimal substructure assumption” is broken, that is, the edge costs depend on each other.

In all of these methods, a best nominal path is computed off-line, which is fixed regardless of the process and sensing noises in the execution phase. Unless the planning domain is small [33], in case of large disturbances, replanning in these methods is computationally very expensive procedure because all of the costs along the path have to be reproduced for the new initial belief. BRM significantly ameliorates this high cost using covariance factorization techniques; however, this still does not satisfy the “optimal substructure assumption,” and thus for a new query from a new initial point, the search algorithm has to be rerun again.

Although the ultimate goal of planning under uncertainty (solving POMDPs) is finding the best policy that generates the optimal actions as a function of occurred noises in execution phase, mentioned methods only return a best nominal path, which is fixed regardless of the process and sensing noises in the execution phase. Thus, to cope with large disturbances, replanning has to be done. However, the main problem with replanning is that the constructed planning tree depends on the initial belief such that all computations needed to construct the tree, that is, to predict the future costs, has to be reproduced for a query from a new starting initial belief.

Thus, online replanning can be done only if the planning domain is small (e.g., [33]) or if the planning horizon is short, such as receding horizon control-based (RHC-based) approaches. The method proposed by Toit and Burdick [38] is an RHC-based method, where the nominal path is updated dynamically over an N -step horizon. PUMA proposed by He et al. [39] also is an RHC-based framework based, where instead of a single action, a sequence of actions (macro-action) is selected at every decision stage. However, these methods entail repeatedly solving open-loop optimal control problems at every time step, which is computationally very expensive as the preceding computations cannot be reused for the new queries from the new initial point.

In FIRM, however, the best feedback policy, that is, a mapping from belief space to actions, is computed off-line, which is a main goal of planning under uncertainty (POMDPs) (e.g., [40] also computes the optimal policy rather than the optimal nominal controls). Moreover, in FIRM the optimal substructure

assumption holds, and as a result, the costs of different edges on the roadmap are independent of each other. Therefore, the roadmap is independent of initial point, and in replannings from a new initial point, the computations need not to be reproduced.

In the methods that account for sensing uncertainty, the state has to be estimated based on measurements. To handle unknown future measurements in the planning stage, methods [32, 33, 36–38] consider only the maximum likelihood (ML) observation sequence to predict the estimation performance. In contrast, FIRM takes all possible future observations into account in planning. References [34] and [35] also consider all possible future observations.

In the presence of obstacles, because of the dependency of collision events in different time steps, it is a burdensome task to include the collision probabilities in planning. Thus, the methods such as [32, 34, 35, 38] design some safety measures to account for obstacles in planning. A problem with some of these collision probability measures is that they are built on the assumption that the collision probability at different stages along the path is independent of each other, which is generally not true and may lead to very conservative plans. However, in FIRM, collision probabilities can be computed and seamlessly incorporated in planning stage, without such an assumption.

4. RELATED WORK: PROBABILISTIC COMPLETENESS

Because of the success of sampling-based methods in many practical planning problems, many researchers have investigated the theoretical basis for this success. However, almost all of these investigations have been done for algorithms that are designed for planning in the absence of uncertainty. The literature in this direction falls into two categories: path isolation-based methods and space covering-based methods.

Path Isolation-Based Analysis

In this approach, one path is chosen, and it is tiled with some sets such as ε -balls in [41] or sets with arbitrary shapes but strictly positive measure in [42]. Then the success probability is analyzed by investigating the probability of sampling in each of the sets that tile the given path, in the obstacle-free space. Methods in [41–44] are among the methods that perform path isolation-based analysis of planning algorithm.

Space-Covering-Based Analysis

In the space-covering-based analysis approach, the adequate number of sampled points to find a successful path is expressed in terms of a parameter ε , which is a property of the environment. A space is ε -good if every point in the state space can at least be connected to an ε fraction of the space using local planners. Methods [45] and [46] are among these methods.

These methods are developed for the case that the solution of a planning algorithm is a path. However, in the presence of uncertainty, the concept of “successful path” is no longer meaningful because on a given path different policies may result in different success probabilities, some interpreted as successful, some not. Thus, because the planning algorithm returns a policy instead of a path, the success has to be defined for a policy. In this chapter we are also aiming in extending these concepts to probabilistic spaces, that is, to sampling-based feedback motion planning under uncertainty. We define the concepts of successful policy, probabilistic completeness under uncertainty, formulate them rigorously, and analyze our methods accordingly.

5. SUMMARY

The work just mentioned involved attempts to solve the motion-planning problem in the presence of uncertain maps, process uncertainty, and sensing uncertainty. In this chapter we address motion-planning problems in the presence of process uncertainty and stochastic maps and finally sensing uncertainty.

This chapter addresses the just-mentioned open issues related to algorithms attempting to solve motion planning in the presence of process uncertainty and map uncertainty and sensing uncertainty. In Sec. II we state the mathematical formulation required to solve this problem. In Sec. III we present a generic solution framework for solving the motion-planning problem in the presence of uncertainty. In the subsequent sections we apply this generic solution methodology to the motion-planning problem with incorporating different uncertainties (first, process uncertainty and then sensing along with process uncertainty) in the planning stage. Then, in Sec. IV we propose a solution methodology that will present a structured way of addressing process and map uncertainty, incorporating the uncertainties in a robust fashion in the planning stage, addressing the motion planning in continuous state and control spaces, and providing performance guarantees. In Sec. V, we present the solution methodology to solve motion-planning problems in the presence of motion and sensing uncertainty.

II. MATHEMATICAL FORMULATION FOR DECISION MAKING UNDER UNCERTAINTY

Mainly, uncertainty in planning originates from the lack of exact knowledge on robot's motion model, robot's sensing model, and environment model, which are referred to as motion uncertainty and sensing uncertainty, but some of the concepts are extendible to the problems with a map uncertainty. The MDP problem and POMDP are the most general formulations, respectively, for the planning problem under motion uncertainty and for the planning problem under both motion and sensing uncertainty.

While in the deterministic setting, we seek an optimal path as the solution of the motion-planning problem; in the stochastic setting we seek an optimal feedback (mapping) π as the solution of the motion-planning problem. In the case of MDP, π is a mapping from the state space to the control space. In the case of POMDP, π is a mapping from the belief space to the control space.

A. MDP

MDP is the most general formulation for the motion-planning problem with process uncertainty. An MDP can be characterized using the following parameters:

1. \mathbb{X} is the state space of the problem. State $x \in \mathbb{X}$ encodes all information needed for decision making at a specific time instant. The state space in our problem is continuous.
2. \mathbb{U} is the control space of the problem, containing all possible control inputs, $u \in \mathbb{U}$. Control space \mathbb{U} can also be continuous. Also, $u_{0:k} : \{u_0, u_1, \dots, u_k\}$ is the control history up to step k .
3. Current state is fully observable.
4. Here $x_{k+1} = f(x_k, u_k, w_k)$ is the state (motion) dynamics, where w_k is the process (motion) noise at time step k .
5. Also, $p(x'|x, u) : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \mapsto \mathbb{R}_{\geq 0}$ is the state transition pdf that encodes the probability density of transition from state x to state x' under the control u . This pdf is also known as the system's dynamics model.
6. Here $c(x, u) : \mathbb{X} \times \mathbb{U} \mapsto \mathbb{R}$ is the cost of taking control u at state x .
7. The $\pi() : \mathbb{X} \mapsto \mathbb{U}$ is the solution of MDP, which is a mapping (feedback) that assigns an optimal control action for every state in state space. It is optimal in the sense that it minimizes the cost-to-go from every state in the state space.

$$\pi = \arg \min_{\Pi} \sum_{k=0}^{\infty} \mathbb{E}\{c[x_k, \pi(x_k)]\} \quad (1)$$

$$s.t. \ x_{k+1} = f(x_k, u_k, w_k) \quad (2)$$

8. Π is the set of admissible policies.
9. $J() : \mathbb{X} \mapsto \mathbb{R}$ is called the optimal cost-to-go (or value) function, which assigns the optimal cost-to-go for every state in state space. It is well known that the optimal cost-to-go is obtained by solving the following stationary dynamic programming (DP) equation for all x on the state-space \mathbb{X} [10]. Subsequently, the solution of MDP, that is, π can be compared as a function that

returns the argument of this minimization, that is, returns the optimal action at every state.

$$J(x) = \min_u \{c(x, u) + \int_{\mathbb{X}} p(x'|x, u)J(x') dx'\} \quad (3a)$$

$$\pi(x) = \arg \min_u \{c(x, u) + \int_{\mathbb{X}} p(x'|x, u)J(x') dx'\} \quad (3b)$$

However, it is well known that the preceding DP equation is exceedingly difficult to solve because it is defined over the whole state space and suffers from the curse of dimensionality [8]. In GPRM, we aim to find an arbitrary good approximation to the solution of the original MDP over finite subsets of state spaces. GPRM is constructed based on sampling-based methods, originally designed for motion planning in deterministic systems, and indeed aims at sampling in the state space.

B. POMDP

As mentioned, POMDP is the most general formulation for the planning problem under process (motion) uncertainty and imperfect state information (sensing uncertainty). POMDPs are introduced in [47–49]. In the following, we itemize the elements that one has to deal with in solving a POMDP problem and then present a form of POMDP formulation, which is known as belief MDP problem [10, 11].

1. \mathbb{X} is the state space of the problem. State $x \in \mathbb{X}$ encodes all information needed for decision making at a specific time instant. It is worth noting that the state space in our problem is continuous.
2. \mathbb{U} is the control space of the problem, containing all possible control inputs, $u \in \mathbb{U}$. Control space \mathbb{U} can also be continuous. Also, $u_{0:k} : \{u_0, u_1, \dots, u_k\}$ is the control history up to step k .
3. \mathbb{Z} is the observation space of the problem, containing all possible observations, $z \in \mathbb{Z}$. Observation space \mathbb{Z} can also be continuous. Also, $z_{0:k} = \{z_0, z_1, \dots, z_k\}$ is the control history up to step k .
4. Here $x_{k+1} = f(x_k, u_k, w_k)$ is the state (motion) dynamics, where w_k is the process (motion) noise at time step k .
5. The $p(x'|x, u) : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \mapsto \mathbb{R}_{\geq 0}$ is the state transition pdf that encodes the probability density of transition from state x to state x' under the control u . This pdf is also known as the system's dynamics model.
6. Here $b_k : \mathbb{X} \times \mathbb{Z}^k \times \mathbb{U}^{k-1} \mapsto \mathbb{R}_{\geq 0}$ is the belief at the k th step, which is the pdf of system state conditioned on the obtained information (measurement and controls) up to k th time step, that is, $b_k = p(x_k | z_{0:k}; u_{0:k-1})$.

7. \mathbb{B} is the belief space of the problem, containing all possible beliefs $b \in \mathbb{B}$.
8. The $b_{k+1} = \tau(b_k, u_k, z_{k+1})$ is the belief dynamics. This dynamics can be derived using Bayes rule and the law of total probability [10, 11].
9. Here $p(b'|b, u): \mathbb{B} \times \mathbb{U} \times \mathbb{B} \mapsto \mathbb{R}_{\geq 0}$ is the belief transition pdf that encodes the probability density of transition from belief b to belief b' under the control u .
10. The $c(b, u): \mathbb{B} \times \mathbb{U} \mapsto \mathbb{R}$ is the cost of taking control u at belief b .
11. Also, $\pi(\cdot): \mathbb{B} \mapsto \mathbb{U}$ is the solution of POMDP, which is a mapping (feedback) that assigns an optimal control action for every belief in belief space. It is optimal in the sense that it minimizes the cost-to-go from every belief in the space.

$$\pi = \arg \min_{\Pi} \sum_{k=0}^{\infty} \mathbb{E}\{c[b_k, \pi(b_k)]\} \quad (4)$$

$$s.t. \ b_{k+1} = \tau(b_k, u_k, z_{k+1}) \quad (5)$$

This form of POMDP is also known as *belief*-MDP problem [10, 11] because indeed it is an MDP over belief space.

12. Π is the set of admissible policies.
13. $J(\cdot): \mathbb{B} \mapsto \mathbb{R}$ is called the optimal cost-to-go (or value) function that assigns the optimal cost-to-go for every belief in belief space. It is well known that the optimal cost-to-go is obtained by solving the following stationary DP equation for all b on the belief space \mathbb{B} [10, 11]. Subsequently, the solution of POMDP, that is, π can be compared as a function that returns the argument of this minimization, returns the optimal action at every belief.

$$J(b) = \min_u \{c(b, u) + \int_{\mathbb{B}} p(b'|b, u) J(b') db'\} \quad (6a)$$

$$\pi(b) = \arg \min_u \{c(b, u) + \int_{\mathbb{B}} p(b'|p, u) J(b') db'\} \quad (6b)$$

However, it is well known that the preceding DP equation is exceedingly difficult to solve because it is defined over the whole belief space and suffers from the curse of history [50]. In FIRM, we aim to find an arbitrarily good approximation to the solution of the original POMDP over finite subsets of belief space. Doing so, in FIRM, we end up with a tractable MDP, the so-called FIRM MDP, as opposed to the intractable original POMDP. FIRM is constructed based on sampling-based methods, originally designed for motion planning in deterministic systems and indeed aims in sampling in the belief space. Doing so, it ameliorates the difficulty of dealing with continuous spaces and also breaks the curse of history in POMDPs [50].

The main tool in breaking the curse of history is the local controllers used in these algorithms' construction. Local controllers are served to ensure that at decision-making stages we reach the nodes. The power of the local feedback controllers resides in the fact that each local controller is a sequence of closed-loop policies (macro-policies), as opposed to the macro-actions, which is a sequence of actions, that is, a sequence of open-loop policies.

III. GENERIC FRAMEWORK FOR SAMPLING-BASED FEEDBACK MOTION PLANNERS

In this section, we describe the elements required to construct a sampling-based feedback motion-planning general framework. Then, we show that the computationally tractable MDP for these sampling-based feedback motion planners (SFMP) serves as an arbitrarily good approximation of intractable original decision processes (MDP or POMDP). We discuss the approximation first in obstacle-free case, and then we add the obstacles to the planning framework. We discuss the quality of solution obtained, that is, the success probability, and finally provide a generic algorithm for planning with the SFMP.

A. PRELIMINARIES

1. FEEDBACK CONTROLLERS IN PARTICULAR SPACES

The decision-making process is performed based on the available information at that stage. In fully observable environments, we have access to system state, and the decision making is performed based on system state. In a partially observable system, we have access to the system's belief (pdf over state), and thus, decision making is performed based on system's belief. Thus, a feedback controller in a given space, say \mathbb{S} , is a mapping from the space to the control space, that is, $\mu(): \mathbb{S} \mapsto \mathbb{U}$. The space \mathbb{S} for fully observable system is $\mathbb{S} = \mathbb{X}$ (state space) and partially observable system is $\mathbb{S} = \mathbb{B}$ (belief space). Generating controls based on a given controller $\mu()$, the state evolves according to a Markov chain whose one-step transition density function is denoted by $p_1^\mu(s'|s) := p[s'|s, \mu(s)]$. Thus, the feedback controller μ essentially induces the Markov chain $p_1^\mu(s'|s)$ over the given space \mathbb{S} . In a similar way, $\mathbb{P}(\cdot | s, \mu): \mathcal{B}_{\mathbb{S}} \mapsto [0, 1]$ is defined as the probability measure over the given space, induced by the local controller μ after one step, starting from the state s . Set $\mathcal{B}_{\mathbb{S}}$ as the Borel sigma-algebra of the space \mathbb{S} .

2. STOPPING REGION

Let $S \subset \mathbb{S}$ be a stopping region of controller μ , if the controller stops the system as the state reaches the region S , that is, $\mathbb{P}_1(s|s, \mu) = 1, \forall s \in S$. In a similar way, $\mathbb{P}_n(\cdot | s, \mu): \mathcal{B}_{\mathbb{S}} \mapsto [0, 1]$ for a controller with stopping region S is defined as the probability measure over \mathbb{S} , induced by the local controller μ after n steps, starting

from the state s and conditioned on the fact that state process has not stopped yet, that is, conditioned on $s_k \notin S, \forall k = 1, \dots, n - 1$.

3. PROBABILITY OF LANDING IN STOPPING REGION

Consider the controller μ that starts executing from state s and stops executing when the state enters region S . Thus, we can define $p(s'|s, \mu)$ as the pdf over the state space when the controller μ , invoked at s , stops executing. Similarly, $\mathbb{P}(\cdot | s, \mu)$ represents the probability measure induced at s when controller stops executing. Thus, the probability of landing in stopping region S in a finite time is $\mathbb{P}(S|s, \mu)$, which is computed as

$$\mathbb{P}(S|s, \mu) = \sum_{n=0}^{\infty} \mathbb{P}_n(S|s, \mu) \quad (7)$$

4. REACHABILITY AND PROPER FEEDBACK POLICY

The stopping region S is reachable under controller μ from s in a finite time if $\mathbb{P}(S|s, \mu) = 1$. The controller μ is called proper with respect to a stopping region S , over the region S_0 , if for all $s \in S_0$, we have $\mathbb{P}(S|s, \mu) = 1$. This can be considered as an extension to the “proper policy” definition in [10]. In this case, the controller-stopping region pair, that is, (μ, S) , is called a proper pair over the region S_0 . The largest S_0 , that is, the set of all states from which the stopping region S is reachable under μ , is called the effective region of pair (μ, S) and is shown by $\mathbb{S}_{(\mu, S)}$. In other words,

$$\mathbb{S}_{(\mu, S)} = \{s \in \mathbb{S} : \mathbb{P}(S|s, \mu) = 1\}, \quad \text{if } \mathbb{S}_{\text{free}} = \mathbb{S} \quad (8)$$

It is worth noting that $S \subset \mathbb{S}_{(\mu, S)}$.

B. OBSTACLE-FREE CONSTRUCTION

In this section, we assume there is no obstacle (constraints) in the state space. As we will see in the next sections, in both state space and belief space the decision-making problem in its most general form is formulated as an MDP problem. When $\mathbb{S} = \mathbb{X}$, we have an MDP in state space. When $\mathbb{S} = \mathbb{B}$, we have an MDP in belief space, which is equivalent to an POMDP in state space. Dynamic programming (DP) is a stochastic optimization algorithm, through which the cost-to-go can be computed for the MDP problem. It is well known that the DP equation is not tractable over the continuous spaces. Thus, inspired by sampling-based motion planning methods, in SFMP we aim to sample the space \mathbb{S} and compute the cost-to-go of these sampled points (nodes) instead of cost-to-go for all points in the space \mathbb{S} . Then, we show that this sampling in space \mathbb{S} leads to a tractable MDP problem that arbitrarily well approximates the original intractable MDP problem.

1. UNDERLYING PRM FOR SFMP

We construct a PRM in the state space with the i th node denoted by $v_i \in \mathbb{X}_{\text{free}}$ and the (i, j) th edge with e_{ij} , which connects v_i to v_j in \mathbb{X}_{free} . We denote the set of nodes and edges of the PRM by $\mathcal{V} = \{v_i\}$ and $\mathcal{E} = \{e_{ij}\}$. We assume that \mathcal{V} includes the goal node into whose vicinity we want to transfer the robot.

2. NODES OF SFMP

Nodes (S_i) in general are disjoint regions/sets of the given space (\mathbb{S}), that is, $S_i \subseteq \mathbb{S}$. Note that $S_i \cap S_j = \emptyset$. Also, these nodes are measurable, that is, $S_i \in \mathcal{B}_{\mathbb{S}}$. The set of all nodes in this space is denoted by $\mathbb{V} = \{S_i\}_{i=1}^{N_v}$.

3. LOCAL CONTROLLERS

Corresponding to every PRM edge e_{ij} , we design a local controller μ^{ij} , such that $\mu^{ij}(\cdot; e_{ij}): \mathbb{S} \rightarrow \mathbb{U}$ is a feedback controller, parameterized by the (i, j) th PRM edge e_{ij} . We denote the set of local controllers $\mathbb{M} = \{\mu^{ij}\}$. The role of the (i, j) th local controller is to take the robot's state s from S_i to S_j . It is a mapping from the given space to the control space that generates control to ensure reaching the region S_j . Thus, the local controller μ^{ij} has to be a proper policy with respect to the region S_j over the initial region S_i as stated in the following assumption.

Assumption 1. We assume that for every edge e_{ij} in PRM, there exists a controller μ^{ij} such that the pair (μ^{ij}, S_j) is a proper pair over the region S_i , $\forall s \in S_i$ where we have $\mathbb{P}(S_j | s, \mu^{ij}) = 1$, in the absence of obstacles. In other words, we assume that in the obstacle-free environment μ^{ij} can drive the state from S_i into S_j in a finite time with probability 1.

4. PLANNING PROBLEM

The planning problem is to transfer the robot's state into the region S_{goal} corresponding to the goal node v_{goal} , with probability 1, following which the system can remain there without incurring any further cost. This is also known as a stochastic shortest path problem [1].

5. ROADMAP OF LOCAL CONTROLLERS

To construct a set of admissible policies Π on which the stochastic optimization using DP can be performed, we exploit the ideas in sampling-based methods. Similar to the PRM, in which the path is constructed as a concatenation of edges on the roadmap, here in SFMP, the policy is constructed by the concatenation of the local policies. Similarly, at each node S_i the set of local controllers that can be invoked is $\mathbb{M}(i) = \{\mu^{ij} \in \mathbb{M} | e_{ij} \in \mathcal{E}\}$. However, by this construction we

still perform planning on the continuous spaces, and we do not discretize the control space.

6. LOCAL CONTROLLERS VS MACRO-ACTIONS

By macro-action here, we mean sequence of controls (actions) [39, 51]. In other words, a macro-action is a sequence of open-loop policies. Local controllers are not a sequence of open-loop actions (macro-actions), but rather they are a sequence of policies (macro-policies); each is a mapping from space \mathbb{S} to the continuous control space \mathbb{U} . Using macro-actions is an open-loop policy that cannot compensate the state deviation from the planned path; however, under local-controllers (macro-policies) the effect of noise can be compensated due to the feedback nature of the controllers, and thus state can be steered towards a stopping region.

7. PROBABILISTIC COMPLETENESS

Also, it can be proved that this construction is probabilistically complete, that is, increasing the number of nodes and local controllers, we can reach a successful policy in the admissible class of policies, if one exists. We discuss on probabilistic completeness later in detail (see Sec. V.D.).

8. INDUCED MDP BY SFMP

According to this construction, the original MDP on the continuous space \mathbb{S} is reduced to an SMDP on \mathbb{S} . The modified DP formulation for this SMDP is given by

$$J(s) = \min_{\mu^j \in \mathbb{M}(i)} C(s, \mu^j) + \int_{S_j} p(s'|s, \mu^j) J(s') ds' \quad \forall s \in S_i, \forall i \quad (9)$$

The integration over the continuous space is reduced to integration over S_j as shown in Eq. (9), because according to Eq. (1), when μ^j stops executing, the state is in S_j with probability 1. Also in the Eq. (9), $C(s, \mu^j) : \mathbb{S} \times \mathbb{M} \mapsto \mathbb{R}_{\geq 0}$ represents the expected cost of invoking the local controller $\mu^j()$ at the state s until the local controller stops executing. Mathematically,

$$C(s, \mu^j) = \sum_{t=0}^{\mathcal{T}^j} c[s_t, \mu^j(s_t) | s_0 = s] \quad (10)$$

$$\mathcal{T}^j(s) = \min_t \{t | s_t \in S_j, s_0 = s\} \quad (11)$$

where \mathcal{T}^j is a random stopping time denoting the time at which the state enters the node S_j under the controller μ^j .

The modified DP, Eq. (9), though computationally more tractable than the original MDP on continuous space is defined as on the continuous neighborhoods S_i and thus still formidable to solve. However, for sufficiently small S_i , the cost-to-go of all states in S_i is approximately equal. A similar statement holds for the incremental cost. Thus, we can define the cost-to-go and incremental cost for a node rather than a state. In other words, we define the cost-to-go $J^g : \mathbb{V} \mapsto \mathbb{R}$ and $C^g : \mathbb{V} \times \mathbb{M} \mapsto \mathbb{R}$ on graph nodes through the following piecewise constant approximation, for sufficiently small S_i :

$$\forall s \in S_i, \forall i, j \begin{cases} J^g(S_i) & := J(s_c^i) \approx J(s) \\ C^g(S_i, \mu^{ij}) & := C(s_c^i, \mu^{ij}) \approx C(s, \mu^{ij}) \\ \mathbb{P}^g(\cdot | S_i, \mu^{ij}) & := \mathbb{P}(\cdot | s_c^i, \mu^{ij}) \approx \mathbb{P}(\cdot | s, \mu^{ij}) \end{cases} \quad (12)$$

where s_c^i is a point in region S_i , for example, its center if the S_i is considered as a ball. Given this approximation, the DP in terms of graph nodes (S_i) in Eq. (9) becomes

$$\begin{aligned} J^g(S_i) &= \min_{\mu^{ij} \in \mathbb{M}(i)} C^g(S_i, \mu^{ij}) + \int_{S_j} p(s' | s, \mu^{ij}) J^g(S_j) ds', & \forall s \in S_i, \forall i, j \\ &= \min_{\mu^{ij} \in \mathbb{M}(i)} C^g(S_i, \mu^{ij}) + J^g(S_j) \int_{S_j} p(s' | s, \mu^{ij}) ds', & \forall s \in S_i, \forall i, j \\ &= \min_{\mu^{ij} \in \mathbb{M}(i)} C^g(S_i, \mu^{ij}) + J^g(S_j) \mathbb{P}(S_j | s, \mu^{ij}), & \forall s \in S_i, \forall i, j \\ &\approx \min_{\mu^{ij} \in \mathbb{M}(i)} C^g(S_i, \mu^{ij}) + \mathbb{P}^g(S_j | S_i, \mu^{ij}) J^g(S_j), & \forall i, j \end{aligned} \quad (13)$$

Note that in the absence of obstacles assumption (1) implies that $\mathbb{P}^g(S_j | S_i, \mu^{ij}) = 1$.

Equation (13) is an arbitrarily accurate approximation to the original DP equation (9) given that the functions $C(s, \mu)$ and $P(S|s, \mu)$ are smooth with respect to their arguments (i.e., at least continuous) and given that the nodes S_i are sufficiently small. The approximation essentially states that any state in the region S_i is represented by s_c^i for the purpose of decision making. In other words, the approximation $\pi^g(S_i) := \pi(s_c^i) \approx \pi(s), \forall s \in S_i, \forall i$ is also made, which leads to

$$J^g(S_i) = \min_{\mu^{ij} \in \mathbb{M}(i)} C^g(S_i, \mu^{ij}) + \mathbb{P}^g(S_j | S_i, \mu^{ij}) J^g(S_j), \quad \forall i, j \quad (14a)$$

$$\pi^g(S_i) = \arg \min_{\mu^{ij} \in \mathbb{M}(i)} C^g(S_i, \mu^{ij}) + \mathbb{P}^g(S_j | S_i, \mu^{ij}) J^g(S_j), \quad \forall i, j \quad (14b)$$

Thus, the original MDP over the entire continuous space \mathbb{S} becomes a finite N -state SMDP as in Eq. (14) defined on the finite set of nodes $\mathbb{V} = \{S_i\}_{i=1}^{N_v}$. It is worth noting that $J^g(\cdot) : \mathbb{V} \mapsto \mathbb{R}$ is the cost-to-go function over the nodes, which assigns a cost-to-go for every node S_i , and the mapping $\pi^g(\cdot) : \mathbb{V} \rightarrow \mathbb{M}$ is a

mapping over the graph, from the set of nodes into the set of local controllers that computes the optimal best local controller that has to be taken at each node.

Given $C^g(S, \mu)$ for all (S, μ) pairs, the DP as in Eq. (14) can be solved *off-line* using standard DP techniques such as value/policy iteration to yield a feedback policy π^g over the nodes S_i .

9. OVERALL POLICY π

Computing π^g , we can compute the overall feedback π generated using this framework, by combining the policy π^g on the graph and the local controllers μ^{ij} . When a local controller μ is chosen using π^g , the local controller starts generating the controls based on the current state at each time step, until the state reaches the corresponding stopping region, denoted by $S(\mu)$. For example, if the controller μ^{ij} is chosen, the stopping region is S_j , that is, $S_j = S(\mu^{ij})$. Once the state enters the stopping region $S(\mu)$, the higher-level decision making is performed over graph by π^g that chooses the next local controller. This hybrid policy is stated as follows:

$$\begin{aligned} \pi : \mathbb{S} &\rightarrow \mathbb{U} \\ u_k = \pi(s_k) &= \begin{cases} \mu_k(s_k), & \pi^g[S(\mu_{k-1})], & \text{if } s_k \in S(\mu_{k-1}) \\ \mu_k(s_k), & \mu_{k-1}, & \text{if } s_k \notin S(\mu_{k-1}) \end{cases} \end{aligned} \quad (15)$$

where $\mu_k \in \mathbb{M}$ denotes the active local controller at time step k . If the initial state is s_0 , the initial local controller is

$$\mu_0() = \begin{cases} \arg \min_{\mu^{ij} \in \mathbb{M}} C(s_0, \mu^{ij}) + \mathbb{P}(S_j | s_0, \mu^{ij}) J^g(S_j), \forall i, j, & \text{if } s \notin \bigcup_m S_m \\ \pi^g(S_l) & \text{if } \exists l, s \in S_l \end{cases} \quad (16)$$

C. INCORPORATING OBSTACLES

In the presence of obstacles, we cannot ensure that the local controller $\mu^{ij}()$ can drive any $s \in S_i$ into S_j with probability 1. Instead, we have to specify the failure probabilities that the robot collides with an obstacle. Let us denote the failure set on \mathbb{X} by F (i.e., $F = \mathbb{X} - \mathbb{X}_{\text{free}}$).

Let us generalize the $\mathbb{P} : \mathcal{B}_{\mathbb{B}} \rightarrow [0, 1]$ to $\mathbb{P} : \{\mathcal{B}_{\mathbb{S}} \cup F\} \rightarrow [0, 1]$ that also measure the failure probability, such that $\mathbb{P}(F | s, \mu^{ij})$ denote the probability that under local controller μ^{ij} the system enters the failure set F before it enters the region S_j , given that the initial state is s . Similarly, we generalize \mathbb{P}^g such that $\mathbb{P}^g(F | S_i, \mu) := \mathbb{P}(F | s_c^i, \mu)$. Again, given function $\mathbb{P}(\cdot | s, \mu)$ is smooth and given that the sets S_j are suitably small, we can make the approximation $\mathbb{P}^g(\cdot | S_i, \mu) := \mathbb{P}(\cdot | s_c^i, \mu) \approx \mathbb{P}(\cdot | s, \mu)$ for all $s \in S_i$ and for all i . Finally, we generalize the cost-to-go function by adding F to its input set, that is, $J^g : \mathbb{V}, F \rightarrow \mathbb{R}$, such that $J^g(F)$ is a user-defined suitably high cost for hitting obstacles. Therefore, we can modify Eq. (14) to incorporate obstacles in the state space by repeating

the procedure in the preceding subsection to get the induced MDP by SFMP in presence of obstacles:

$$J^g(S_i) = \min_{\mu^{ij} \in \mathbb{M}(i)} C^g(S_i, \mu^{ij}) + J^g(F) \mathbb{P}^g(F|S_i, \mu^{ij}) + J^g(S_i) \mathbb{P}^g(S_j|S_i, \mu^{ij}) \quad (17a)$$

$$\pi^g(S_i) = \min_{\mu^{ij} \in \mathbb{M}(i)} C^g(S_i, \mu^{ij}) + J^g(F) \mathbb{P}^g(F|S_i, \mu^{ij}) + J^g(S_i) \mathbb{P}^g(S_j|S_i, \mu^{ij}) \quad (17b)$$

It is assumed that the system can enter the goal region or the failure set and remains there subsequently without incurring any additional cost. Thus, all that is required to solve the preceding DP equation (17) are the values of the costs $C^g(S_i, \mu^{ij})$ and transition probability functions $\mathbb{P}^g(\cdot | S_i, \mu^{ij})$. Thus, the main difference from the obstacle-free case is the addition of a “failure” state to the MDP along with the associated probabilities of failure from the various nodes S_i .

Again, the overall hybrid policy π and initial local controller μ_0 can be computed as follows:

$$\pi: \mathbb{S} \mapsto \mathbb{U}, \quad (18)$$

$$u_k = \pi(s_k) = \begin{cases} \mu_k(s_k), & \pi^g[S(\mu_{k-1})], & \text{if } s_k \in S(\mu_{k-1}) \\ \mu_k(s_k), & \mu_{k-1}, & \text{if } s_k \notin S(\mu_{k-1}) \end{cases}$$

where

$$\mu_0() = \begin{cases} \arg \min_{\mu^{ij} \in \mathbb{M}} C(s_0, \mu^{ij}) + \mathbb{P}(S_j|s_0, \mu^{ij})J^g(S_j) + \mathbb{P}(F|s_0, \mu^{ij})J^g(F), & \forall i, j \\ \pi^g(S_i) \end{cases} \quad (19)$$

D. SUCCESS PROBABILITY

We would also like to quantify the quality of the solution π . To this end, we require the probability of success of the policy π^g at the higher-level Markov chain on S_i given by the feedback policy defined by Eq. (17b) has $N_v + 1$ states $\{\bar{S}_1, \bar{S}_2, \dots, \bar{S}_{N_v+1}\}$, which can be decomposed into three disjoint classes: the goal class, $\bar{S}_1 = S_{\text{goal}}$; the failure class, $\bar{S}_2 = F$; and the transient class $\{\bar{S}_3, \bar{S}_4, \dots, \bar{S}_{N_v+1}\} = \{S_1, S_2, \dots, S_{N_v}\} \setminus S_{\text{goal}}$. The goal and failure classes are absorbing recurrent classes of this Markov chain [52]. As a result, the transition probability matrix of this higher level $N_v + 1$ state Markov chain can be decomposed as follows [53]:

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{\text{goal}} & 0 & 0 \\ 0 & \mathcal{P}_f & 0 \\ \mathcal{R}_{\text{goal}} & \mathcal{R}_f & \mathcal{Q} \end{bmatrix} \quad (20)$$

The (i, j) th component of \mathcal{P} represents the transition probability from \bar{S}_j to \bar{S}_i , that is, $\mathcal{P}[i, j] = \mathbb{P}^g[\bar{S}_i | \bar{S}_j, \pi^g(\bar{S}_j)]$. Moreover, $\mathcal{P}_{\text{goal}} = \mathbb{P}^g[S_1 | S_1, \pi^g(S_1)] = 1$ and $\mathcal{P}_f = \mathbb{P}^g(F | F,) = 1$ because goal and failure classes are the absorbing recurrent classes, that is, the system stops once it reaches the goal or it fails. \mathcal{Q} is a matrix that represents the transition probabilities between nodes S_i in transient class, whose (i, j) th element is $\mathcal{Q}[i, j] = \mathbb{P}^g[\bar{S}_{i+2} | \bar{S}_{j+2}, \pi^g(\bar{S}_{j+2})]$. Vectors $\mathcal{R}_{\text{goal}}$ and \mathcal{R}_f are $(N_v - 1) \times 1$ vectors that represent the probability of the transient nodes $\mathcal{V} \setminus S_{\text{goal}}$ getting absorbed into the goal node and the failure set, respectively, that is, $\mathcal{R}_{\text{goal}}[j] = \mathbb{P}^g[S_1 | \bar{S}_{j+2}, \pi^g(\bar{S}_{j+2})]$ and $\mathcal{R}_f[j] = \mathbb{P}^g[\bar{S}_2 | \bar{S}_{j+2}, \pi^g(\bar{S}_{j+2})]$. Then, it can be shown that the success probability from any desired node $\bar{S}_i \in \mathcal{V} \setminus S_{\text{goal}}$ is given as follows [53]:

$$\mathbb{P}(\text{success} | \bar{S}_i, \pi) := \mathbb{P}(S_{\text{goal}} | \bar{S}_i, \pi) = \Gamma_{i-2}^T (\mathcal{I} - \mathcal{Q})^{-1} \mathcal{R}_{\text{goal}}, \quad \forall i \geq 3 \quad (21)$$

where Γ_i is a column vector with all elements equal to zero but the i th element, which is one, and \mathcal{I} is the identity matrix. Note that the vector $\mathcal{P}^s = (\mathcal{I} - \mathcal{Q})^{-1} \mathcal{R}_{\text{goal}}$ includes the success probability from every node. For the motion-planning problem under uncertainty, we can no longer ensure a solution that will succeed with probability 1; hence, for stochastic environments and motion planning under uncertainty, the solution should maintain a high success probability or should be $\geq p_{\min}$ for some given p_{\min} for the particular environment. Later, we discuss the success probability in more detail.

E. GENERIC ALGORITHMS

The generic algorithms for off-line construction of SFMP and online planning on them are presented in algorithms 1 and 2. The concrete instantiations of these algorithms (for the LQR/feedback-based GPRM and LQG-based FIRM) are given in the subsequent sections.

Algorithm 1. Generic-Construction of Generalized RoadMap for SFMP (Off-line)

1. Construct a PRM with nodes $\mathcal{V} = \{v_j\}$ and edges $\mathcal{E} = \{e_{ij}\}$;
2. For each PRM edge e_{ij} , design a local controller μ^{ij} , and compute its corresponding reachable stopping region S_j (graph node);
3. For each S_k and $\mu^{ij} \in \mathbb{M}(i)$, compute the transition cost $C^g(B_i, \mu^{ij})$ and transition probability $\mathbb{P}^g(S_j | S_i, \mu^{ij})$ and $\mathbb{P}^g(F | S_i, \mu^{ij})$ associated with going from S_i to S_j (more rigorously, associated with taking μ^{ij} at S_i);
4. Solve the tractable MDP in Eq. (17) to compute the feedback π^g over the nodes, and compute π^g accordingly;
5. Ensure success probability $\mathcal{P}^s \geq p_{\min}$ else resample.

Algorithm 2. Generic-Planning (or Replanning) on Generalized RoadMap for SFMP (Online)

1. Given an initial state s_0 , invoke the controller $\mu_0()$ using Eq. (19) to absorb the robot into some node $S_i \in \mathbb{V}$;
2. Given the system is in set S_i , invoke the higher-level feedback policy π^g to choose the lower-level local feedback controller $\mu^{ij}() = \pi^g(S_i)$;
3. Let the local controller $\mu^{ij}()$ execute until absorption into the S_j or failure;
4. Repeat steps 2–3 until absorption into the goal node S_{goal} or failure.

F. DISCUSSION

In summary, in SFMP, we aim to solve the original MDP on \mathbb{S} , corresponding to decision-making problem on a subset of \mathbb{S} . Given the smoothness of cost function and transition probabilities, the solution is arbitrarily close to the solution of original MDP or POMDP over $\cup_m S_m$, which is a subset of \mathbb{S} . The important characteristics of these solution methodologies are that they are solved off-line and thus prove that the online phase of planning (or replanning) is computationally feasible.

IV. SAMPLING-BASED FEEDBACK MOTION PLANNERS WITH PROCESS UNCERTAINTY AND STOCHASTIC MAPS

In this section, the framework developed in the preceding chapter is applied to the motion-planning problem in the presence of process uncertainty and stochastic maps. The generic space \mathbb{S} with state $s \in \mathbb{S}$ is replaced by the state space of the robot, that is, \mathbb{X} and corresponding state $x \in \mathbb{X}$, and the roadmap is built as developed in the preceding section. In this section we do not incorporate sensing uncertainty into the motion-planning problem, and the particular problem will be discussed in the next section, where we use the framework to solve the motion-planning problem in the presence of process as well as sensing uncertainty.

The robotic motion-planning problem can be formulated as a MDP, if the uncertainties in the robot motion and environments can be modeled probabilistically. This MDP becomes infinite dimensional if considered in continuous state and control spaces. Using hierarchical methods, this infinite dimensional MDP can be modeled as SMDP, which involves solving the MDP at some “distinguished states” that we will call *nodes* or *node states*.

A hierarchical-based generalized sampling-based motion planners will be developed (Fig. 1), and details of the development will be discussed in Sec. IV.A. These generalized sampling-based motion planners are generalization of *probabilistic roadmaps* (PRM) and *rapidly exploring random tree* (RRT) and are called *generalized PRM* (GPRM) and *generalized RRT* (GRRT), respectively.

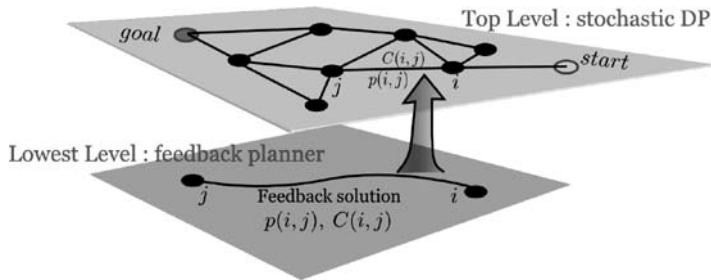


FIG. 1 Depicting hierarchical planning in levels (for single agent).

The solution methodology, as proposed in the preceding section while developing the framework, for these generalized sampling-based motion planners involves the following:

1. Sample configurations (nodes) from the free state-space (\mathbb{X}_{free}); the method of doing this differs in GPRM and GRRT.
2. Find a feedback-based control (μ^{ij}) solution to transition of robot from one node state X_i to another X_j , (the lowest-level planner).
3. Estimate the transition cost $[C(X_i, \mu)]$ and transition probabilities ($\mathbb{P}(X_j|X_i, \mu^{ij})$) for each of these transitions by using the stochastic maps provided and Monte Carlo simulations.
4. Generate a topological graph $\mathcal{G}(\mathbb{V}, E)$ using the sampled nodes ($\mathbb{V} = \{S_i\}_{i=1}^{N_v}$) and the connections ($E = \{e_{ij}\}$) made.
5. Run stochastic DP (the top-level planner), an optimization algorithm over the developed graph to get solution in GPRM.
6. Keep developing the graph until the start configuration is connected to the desired goal configuration in GRRT.
7. Ensure that the probability of the final path achieved either in GPRM or GRRT has probability of success p_s , as developed in Eq. (21) associated with it and it maintains $p_s > p_{\min}$, where p_{\min} is a priori specified for a given environment and is the minimum acceptable success probability for any path solution.

There is a collision detection module that ensures that the sampled configurations q are obstacle free, that is, $p(\mathcal{O}|q) = 0$, where \mathcal{O} represents the obstacles and also the path (the edge on \mathcal{G}) generated by the lowest-level planner (the feedback solution) in between two configurations or nodes that have a $p_{\text{edge}} > 0$. This collision detection module has the stochastic map provided as the input.

A. GENERALIZED SAMPLING-BASED MOTION PLANNERS

In this section, we present the generalized sampling-based motion planners, GPRM and GRRT, that extend the traditional PRM and RRT algorithms to systems in the presence of process uncertainty and stochastic maps. In the following sections, we first formulate the uncertainty models, followed by a detailed description of the generalized algorithms.

1. MODEL

The generalized sampling-based algorithms require an uncertainty model for both the motion of the robot and a model for map uncertainty. In the following discussion, we outline the models that are used in this section.

We assume that the dynamics of the mobile robot are specified by the following white-noise-perturbed stochastic differential equation:

$$\dot{x} = f(x) + g(x)u + h(x)w \quad (22)$$

where x represents the state of the robot, w represents the white-noise perturbation, and u represents the control input to the robot. The aforementioned equation is a nonparametric model of uncertainty in the robot motion model and will be used throughout this section for the lowest-level control law designs.

We assume that the uncertainty in the map is specified through a binary occupancy value $p(O/y)$, that is, the probability that there is an obstacle at the point y in the map. The occupancy values in the map can be considered the output from a mapping algorithm. However, in this chapter we shall not cover the mapping algorithm and assume that a map with binary occupancy values is provided to the planner by some suitable mapping algorithm. The state of the robot consists of $x = (q, \dot{q})$, where q represents the configuration of the robot and \dot{q} represents the generalized velocities. The free region in the map induces a free region in the configuration space, for example, $\mathcal{C}_{\text{free}}$. This case means that any state whose configuration is in $\mathcal{C}_{\text{free}}$ is safe. This condition, in turn, induces a free space in the state space of the robot, for example, $\mathcal{X}_{\text{free}}$. From now on, we will assume that, for GPRM and GRRT, we are sampling equilibrium states, that is, states wherein the velocities are zero, in the free state space $\mathcal{X}_{\text{free}}$.

Furthermore, we shall assume in this section that the state of the robot is perfectly known. The case of imperfect state observation will be considered in the next section of this chapter.

2. GPRM

In motion planning, the objective is to plan the path of a robot from a start state to an end state. PRM attempts to accomplish this condition by the following two approaches: 1) randomly sampling the state space of the robot and 2) connecting every sampled point with its k -nearest neighbors using some local open-loop

planner such as a straight-line planner while checking for collisions with obstacles. The result of PRM is a graph or roadmap on the workspace of the robot that contains the feasible connections between the sampled points in the state space. The problem is solved if there exists a path on the graph that connects the start and the goal states; otherwise, more points are sampled in the state space of the robot until a graph that contains such a feasible path is found. In the case of systems with uncertainty, it may be impossible to find a path that succeeds with probability 1, and, hence, we are interested in finding paths that have a success probability above a prescribed minimum threshold p_{\min} . The algorithm is analyzed and later in this work shown to be probabilistic complete. Hence, if there exists a path with probability of success greater than p_{\min} , the algorithm will find it with probability 1. In situations when the algorithm is applied to maps with no possible solution, the algorithm will stop after a large number of iterations and return failure.

The pseudocode for the GPRM algorithm is shown as follows. As shown in the pseudocode, steps 2, 3 and 5 are different from the traditional PRM algorithm. In the following sections, we discuss these steps of the algorithm in detail.

Algorithm 3. GPRM Algorithm

Date: the start state x_0 , goal state x_g , minimum probability of success p_{\min}

1. Initialize the GPRM with nodes x_0 and x_g ;
2. **for** $p_s > p_{\min}$ **do**
3. Sample equilibrium states in $\mathcal{X}_{\text{free}}$ probabilistically using a uniform distribution;
4. Grow the GPRM by connecting every sampled state in the domain with its k -nearest neighbors using suitable obstacle-free feedback controllers;
5. Evaluate the cost of every connection in the resulting graph using Monte Carlo simulations;
6. Plan on the resulting graph using the evaluated edge cost from step 5;
7. Evaluate the probability of success p_s of the resulting path from step 6, and set $p_s = 0$ if there is no path.

Step 4: Given the robot dynamics as defined in the preceding section and some equilibrium point x_g in the state space of the robot, there exists a feedback controller $u, (x_g)$ such that the robot can be controlled into a neighborhood of the sampled state space x_g , that is, X_g , with some (high) probability, in the presence of the stochastic disturbance forces and in the absence of any obstacles in the map. Note here that the equilibrium state of the robot x_g corresponds to some location in the map that the robot needs to reach. Let X_g denote a neighborhood of the sampled state space x_g . Then, the aforementioned case implies that the probability of the state of the robot $p[x(t)]$ is mostly concentrated in the region X_g as $t \rightarrow \infty$.

Step 5: The feedback controller $\mu^j()$ that we design for controlling the robot from one node X_i to another X_j is for an obstacle-free map, and, hence, there is no guarantee that the controller will succeed in connecting the two nodes in the presence of obstacles. Thus, we need to test the controller through

repeated simulations to evaluate its probability of success. This condition can precisely be stated as follows. Given a start node X_i (at state x_i) and a target node X_j (at state x_j), we may evaluate the probability of success of the local controller $u_i(x_i)$ in connecting the nodes as follows. Recall that we are never sure to be in either node x_i or x_j due to uncertainty in the system. Hence, the feedback controller to control the system from $x_i \rightarrow x_j$ is turned on when the state of the robot enters some prespecified neighborhood of x_i , for example, X_i , and turned off when the state of the robot enters some neighborhood of the node x_j , for example, X_j , at which time the feedback controller, to get it to one of the neighboring nodes of x_j , is switched on. (This situation is shown in Fig. 2.) Let one particular instance of a trajectory, for example, the N th instance, which goes from $X_i \rightarrow X_j$ under the feedback controller $u(\cdot; x_j)$, be $x_0^{(N)}, \dots, x_{t(N)}^{(N)}$, where $t(N)$ denotes the time that the controller terminates. This time $t(N)$ is stopping time and is a random variable because it depends on the particular realization. The probability of success of the N th realization is given by:

$$p_s^{j,(N)} = \{1 - p[O/x_0^{(N)}]\} \cdots \{1 - p[O/x_{t(N)}^{(N)}]\} \quad (23)$$

where, as aforementioned, $p(O/x)$ is the occupancy probability that there is an obstacle at the point x in the state space of the robot. In addition, we can find the cost of the plan from x_i to x_j , $c_{ij}^{(N)} = C(X_i, \mu)$ in terms of physical variables such as fuel and time. Then, if we do repeated simulations, the probability of success and cost of the controller $u(\cdot; x_g)$ in controlling the robot from x_i to x_j can be approximated as

$$p_s^{jj} \approx \frac{1}{M} \sum_{N=1}^M p_s^{j,(N)} \quad (24)$$

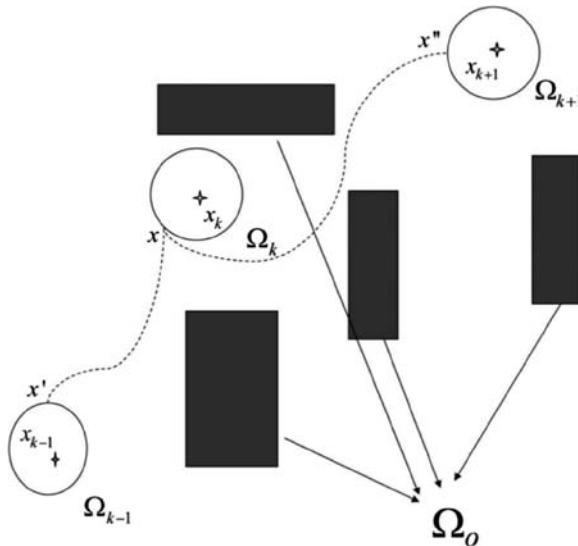


FIG. 2 Illustration of robot motion under the GPRM methodology.

$$c_s^{ij} \approx \frac{1}{M} \sum_{n=1}^M c_s^{ij,(N)} \quad (25)$$

Because of law of large numbers, it follows that as $M \rightarrow \infty$ the aforementioned estimates converge to the true values of the parameters. Given the probability of success of a controller in connecting nodes x_i and x_j and the cost in successfully connecting them, the cost of the edge connecting the nodes x_i and x_j in the graph is given by

$$c^{ij} = p_s^{ij} c_s^{ij} + (1 - p_s^{ij}) c_f \quad (26)$$

where c_f is some heuristically defined, suitable high cost of failure. The aforementioned equation allows the evaluation of the edge costs in the graph that is formed by connecting any node to its k -nearest neighbors.

Step 7: In traditional PRM, if we find a path from the start node to the goal node, the planning problem is solved. However, in the presence of uncertainty, we have to ensure that the probability of success of the path planned on the graph is above the minimum threshold value of p_{\min} . Thus, it is not necessary that, if there is a path from the start node to the goal node, it has the minimum required probability of success. This case has to be tested. Thus, once a minimum cost path is found on the graph according to the edge costs as defined earlier, the probability of success of the individual segments of the path, which, in turn, is from step 5. Thus, if the success probability is higher than the threshold, the planning problem is solved; otherwise, more points have to be sampled in the state space.

Remark 1: The feedback controller is designed for a workspace without any obstacles because otherwise the controller design is very complicated due to the constraints imposed on the robotic system by its workspace. The requirement of feedback controllers that stabilize a system to a given equilibrium point x_g is mild. In fact, for any fully actuated system, linearizing the system about the given equilibrium and designing a linear quadratic (LQ) controller for the linearized system results in such a stabilizing controller, at least locally. For nonholonomic and underactuated systems, in general, such linearized techniques may not provide a stabilizing controller due to Brockett's theorem [54]. In that case, suitable nonlinear controllers may be designed to stabilize the system about a nominal trajectory, which can be obtained using optimal control techniques. In the example section, we use a dynamic feedback linearization-based controller to design stabilizing controllers about equilibrium configurations of a unicycle robot.

Remark 2: The analytical evaluation of the probability of success of an option is difficult because it is equivalent to the "first passage time" problem for a stochastic nonlinear system. In general, Monte Carlo techniques, including sequential Monte Carlo techniques, are the most efficient method for evaluating such probabilities [55]. In this work, we use simple Monte Carlo to evaluate these probabilities. The time of execution of the options $t(\omega)$, where ω is a particular realization of the trajectory under the option, can, in general, be infinite, that is, the option may take an infinite time to terminate. However, the expected time to terminate can be shown to be finite, $E[t] < \infty$ (refer analysis of the more general algorithm in Sec. V.E).

3. GRRT

The traditional RRT algorithm attempts to connect a start point and an end point in the workspace of a robot by growing a tree using a random sampling as follows:

1. Randomly pick a point in the state space of the robot.
2. Find the nearest node on the tree according to some prespecified metric.
3. Connect the nearest node on the tree to the sampled node using some local planner while checking for collision.
4. Add the new node to the tree if the robot does not collide with obstacles.

The tree is grown in this manner until a feasible path is found from the start point to the goal point. Because of uncertainty, it might not be possible to find a path that succeeds in connecting two points with probability 1. Hence, in the current scenario we require that the path has a minimum prespecified probability of success p_{\min} .

We now present the generalized version of the RRT algorithm, that is, GRRT. The pseudocode for the algorithm is presented as follows. (Steps 2–4 of the algorithm are different from the traditional RRT algorithm, and in the following sections we discuss the details about these differences, starting with step 3.)

Algorithm 4 GRRT Algorithm

Data: Start state x_0 , goal node x_g , minimum probability of success p_{\min}

Input: Initialize tree with x_0 , set $p(x_0) = 1$, set $N = 1$

1. **for** tree reached goal node x_g **do**
2. Generate node x_N at random (x_N is an equilibrium state in $\mathcal{X}_{\text{free}}$);
3. Connect x_N to the node x^* on the tree, using local feedback control, that satisfies

$$x^* = \arg \max_i p(x_i)p(x_i, x_N) \quad (27)$$

where $p(x_i, x_N)$ is the probability of successfully transitioning from $x_i \rightarrow x_N$ under the local feedback law;

4. Set $p(x_N) := p(x^*)p(x^*, x_N)$;
5. **if** $p(x_N) > p_{\min}$ **then**
6. add node x_N to the tree with label $p(x_N)$ and set $N = N + 1$
7. **else**
8. Go to step 2
9. **end for**

Step 3: The nodes (or, more precisely, the neighborhoods of the nodes) are connected by local feedback controllers that have been designed using control design techniques and the probability of success of

the controller evaluated as in the GPRM algorithm. The reason that we chose the nodes as in Eq. (27) has to do with the proof of completeness of the resulting algorithm. In fact, the choice can be thought of as the “nearest node” metric that is used to select the node in the tree that is connected to the newly generated node. Hence, x^* as defined in Eq. (27) is the best node in terms of the probability of success of transitioning from $x_0 \rightarrow x^* \rightarrow x_N$, where, as aforementioned, x_0 is the root node.

Steps 4 and 5: Step 4 labels any newly generated node with the “probability of success” of the robot moving from the root node to that particular node. We only want to keep nodes in the tree that have a success probability (of transitioning to it from the root node) more than the threshold of p_{\min} . Hence, we include the tree pruning in step 5. Note that, given the probability of success, $p(x_i, x_j)$ of transitioning from any parent node x_i to its children x_j in the tree, the probability of success of a path x_0, x_1, \dots, x_K is given by $p(x_0, x_1)p(x_1, x_2) \dots p(x_{m-1}, x_m) = p(x_{K-1})p(x_{K-1}, x_K)$ according to the labeling convention that we have used, where, as aforementioned, $p(x)$ represents the probability of successfully transitioning from root node to node x .

GRRT algorithm is also analyzed and later in this work shown to be probabilistic complete. Hence, if there exists a path with probability of success greater than p_{\min} , the algorithm will find it with probability 1. In situations when the algorithm is applied to maps with no possible solution, the algorithm will stop after a large number of iterations and return failure.

Probabilistic completeness for these SFMP in the presence of process uncertainty and stochastic maps, that is, GPRM and GRRT, can be proven in a similar fashion as it will be for the SFMP algorithm FIRM (SFMP in the presence of process and sensing uncertainty) in Sec. V.D.

B. NUMERICAL EXPERIMENTS

In this section, we will detail the application of the generalized sampling-based motion to a fully actuated holonomic point robot and an underactuated nonholonomic unicycle model.

1. FULLY ACTUATED POINT ROBOT

In this section, we apply the generalized planners developed in the preceding section to an idealized holonomic point robot. We will deal with a planner robotic system, but the extension to a three-dimensional system is quite straightforward. Note that the higher-level planning algorithms do not change with the robot model, and only the lowest-level controllers change with different or complicated robot models. The design of local point-to-point feedback controllers for holonomic systems such as the fully actuated point robot considered here is quite trivial and can be done using the standard LQ control theory [1].

The robot motion model was assumed to be:

$$\ddot{q} = u + w \quad (28)$$

where q is the position vector, u are the input forces, and w is a white-noise term that quantifies the uncertainty in the motion model of the robot. We assumed that

the state of the robot could perfectly be sensed. We used several different maps and assumed that the map uncertainty in each case was specified to us using a discrete occupancy-grid (OG) representation, that is, we are given a distribution $p(O/x_{ij})$ that denotes the probability that there is an obstacle in the (i, j) th grid in the map.

The local feedback controllers that connect the sampled points were designed using LQR techniques [1]. The cost function used penalized both the control effort and the state deviation from the goal equilibrium point. The safe recurrent classes X_k around some sampled point x_k were defined to be some ball of radius ϵ , where ϵ was chosen in a heuristic manner. We found the probability of success and the cost of operation, where the cost of operation was quadratic cost, of the feedback controllers that join two nodes on the graph using repeated Monte Carlo simulations according to Eqs. (24) and (25). We have tested the controllers using a naive Monte Carlo method, but for higher dimensional problems very efficient subset simulations technique exist [55], which can be leveraged to find the success probabilities in an efficient manner.

The results of our simulation experiments are shown in Figs. 3–6. Each of these figures presents the performance of the GPRM and GRRT algorithms, along with their traditional counterparts on two different maps. We have performed our simulations on several other maps, but the paucity of space does not allow us to present all of these results. Figures 4a and 6a represent the tree built RRT in the map. In Figs. 4b and 6b, we represent the final nominal (noise-free) path from the start node to the goal node found by RRT with a solid line and the ensemble of trajectories that result due to the process uncertainty in the motion model with a dotted line, along with probability of success of the path. In Figs. 4c and 6c, were present the tree built by GRRT, showing the noise-free paths between nodes. In Figs. 4d and 6d, we represent the nominal noise-free path, along with the ensemble of trajectories due to process uncertainty, as well the probability of success of the path. Similarly, in Figs. 3a and 5a, we show the graph built by PRM and the nominal path and the trajectory ensemble, along with the probability of success of the path. The same data are presented for GPRM in Figs. 3c and 3d and Figs. 5c and 5d. The graphs and trees constructed by GPRM and GRRT are virtual because there is not one single path that connects the nodes but an entire bundle of them between any two nodes because of the uncertainty in the robot motion model. Hence, the connection encodes the feedback controller that joins the two nodes, rather than an actual path that joins them as in the case in traditional PRM or RRT. However, for visual comprehension, we only present the noise-free paths in the GRRT and GPRM tree and graph figures, respectively (see Figs. 3c, 4c, 5c, and 6c). As shown in the figures, the performance of GPRM and GRRT is significantly better than the performance of the traditional PRM and RRT algorithms. For instance, in map 1 the probabilities of success of the GRRT and GPRM algorithms are 100 and 89.68%, respectively, whereas the probabilities of success of the traditional RRT and PRM are 12 and 2%, respectively. A similar observation holds for map 6. Although the nominal noise-free path in

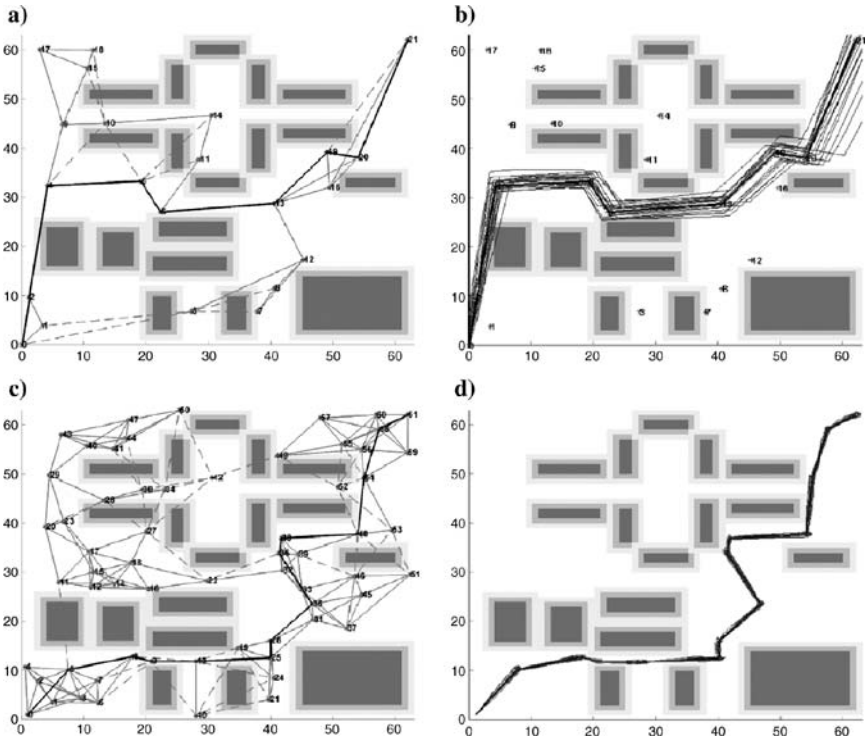


FIG. 3 Comparison of GPRM with traditional PRM: map 1 ($p_s \equiv$ probability of success): a) standard PRM; b) $p_s = 2\%$, trajectories ensemble; c) modified PRM; and d) $p_s = 89.68\%$, trajectories ensemble.

PRM and RRT does not collide with any obstacles, the presence of the noise in the robot motion model leads to collisions in most cases due to the open-loop nature of the plan. This case is clearly shown in Figs. 3b, 4b, 5b, and 6b, where the trajectory ensemble grows in size over time and diverges from the nominal noise-free path and thereby leads to collisions, which were not present in the nominal path. Moreover, it is also shown in the same figures that there is no guarantee that the robot will reach the goal under these plans. In contrast, the robustness that is attained due to feedback can be gauged from Figs. 3d, 4d, 5d, and 6d where the ensemble of trajectories is tightly bundled around the nominal noise-free path due to the presence of feedback. We note here that the very low success probabilities of the traditional methods are due to the complicated nature of the two maps. In similar maps, they have better performance, but the GRRT and GPRM performance is always significantly better. This result is not surprising because the original PRM and RRT algorithms are open-loop planners

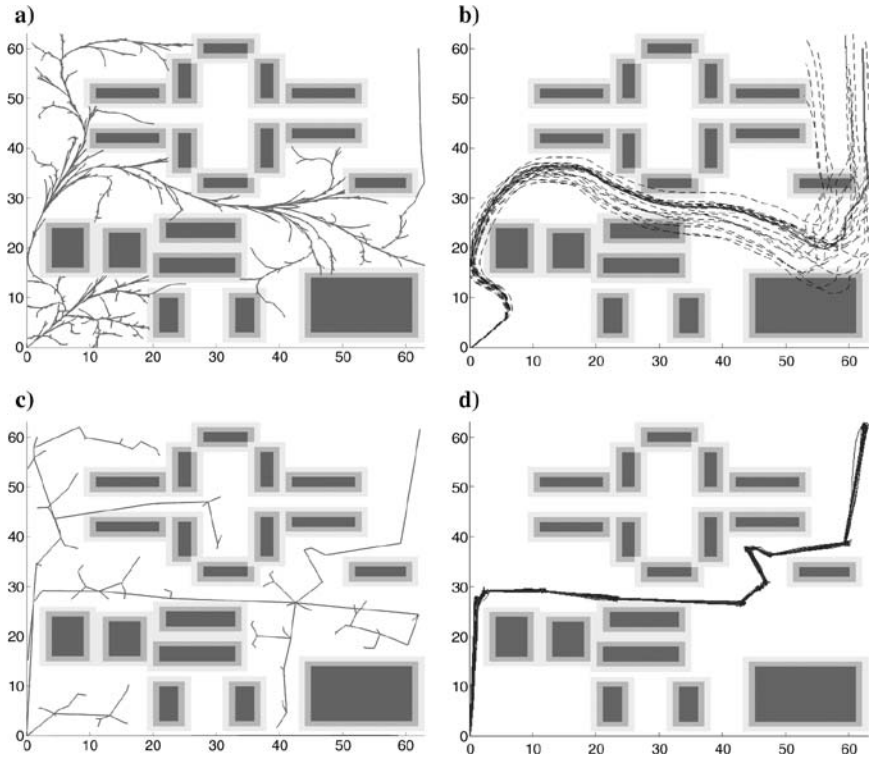


FIG. 4 Comparison of GRRT with traditional RRT: map 1 (p_s = probability of success): a) standard RRT tree; b) $p_s = 12\%$, trajectories ensemble; c) feedback-based RRT tree; and d) $p_s = 100\%$, trajectories ensemble.

and were not developed for uncertain robot models and state spaces. The control in the open-loop planners is a function of time alone and is based on the nominal dynamics (unperturbed dynamics). Thus, in the presence of perturbations or noise, the path, if the robot can substantially deviate, and the nominal performance of the robot cannot be maintained. However, the feedback plans in GRRT and GPRM ensure robustness to such perturbations because the control is a function of the state of the robot (not time), and hence, even when the robot path deviates from the nominal, the control law can still guide it towards the goal. In fact, all other techniques that modify the PRM and RRT to handle uncertainty, such as [19–22], suffer from the exact same problem because the open-loop planners that are designed to handle only stationary map uncertainty and thus, similar to PRM and RR as aforementioned, cannot be robust to the dynamic process uncertainty in the robot motion model.

2. NONHOLONOMIC UNICYCLE ROBOT

In this section, we apply the sampling-based motion planners to the motion planning of a unicycle model whose equations of motion are given by

$$\dot{x} = v \cos \theta + w_x \quad (29)$$

$$\dot{y} = v \sin \theta + w_y \quad (30)$$

$$\dot{\theta} = \omega + w_\theta \quad (31)$$

where (x, y, θ) represents the pose of the robot; the velocity v and the angular velocity ω represent the control inputs to the problem; and w_x , w_y , and w_θ are uncorrelated white-noise terms. We assume that the robot can be approximated by a point. In this case, our sampled poses are in the (x, y, θ) spaces, and the job of the local feedback controllers is to stabilize the robot about any of these equilibrium

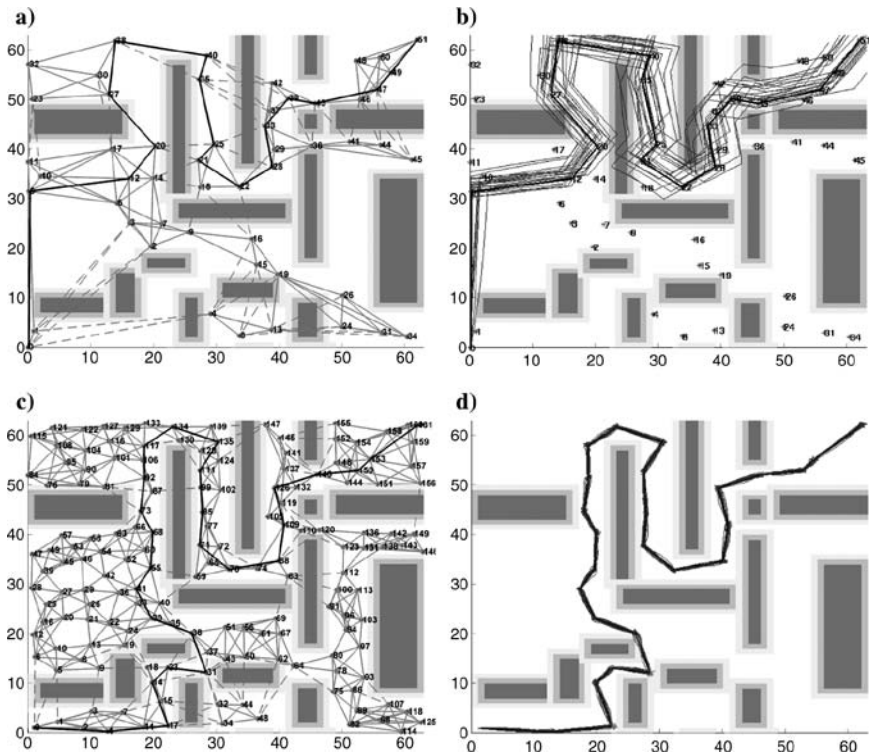


FIG. 5 Comparison of GPRM with traditional PRM: map 6 ($p_s \equiv$ probability of success): a) standard PRM; b) $p_s = 6\%$, trajectories ensemble; c) modified PRM; and d) $p_s = 96.83\%$, trajectories ensemble.

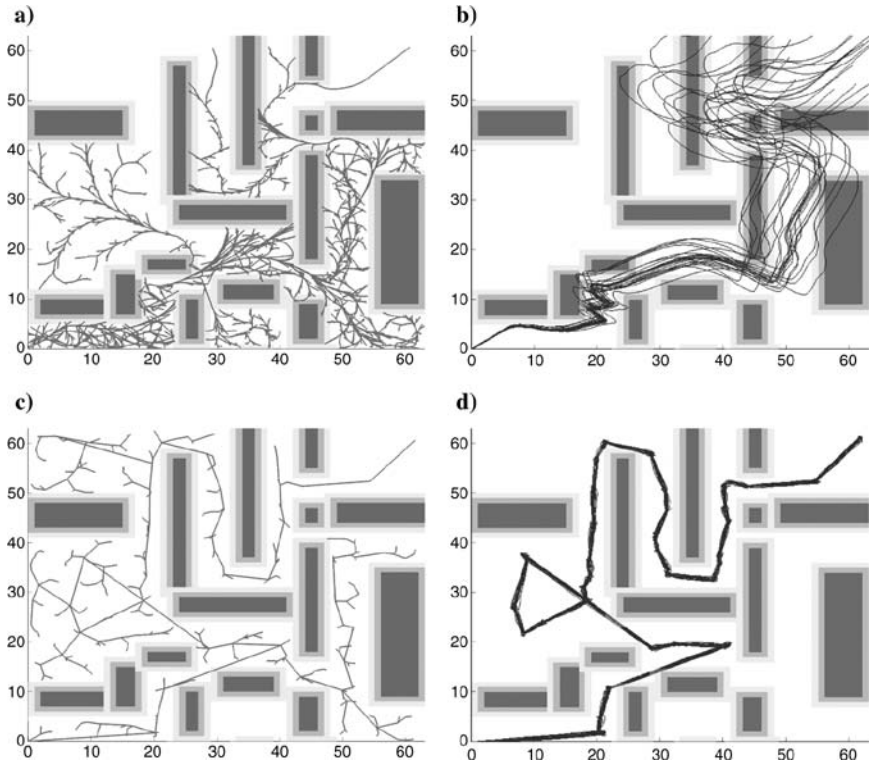


FIG. 6 Comparison of GRRT with traditional RRT: map 6 (p_s = probability of success): a) standard RRT tree; b) $p_s = 0\%$, trajectories ensemble; c) feedback-based RRT tree; and d) $p_s = 87.9\%$, trajectories ensemble.

configurations. Because the unicycle model is a nonholonomic system, the design of feedback controllers that stabilizes the robot about a particular equilibrium configuration is much more involved than in the case of the fully actuated robot considered in the preceding section. In fact, the standard linear control theory cannot be used in the design of feedback controllers for such systems, even locally [56, 57], because it is known from Brockett's theorem that a static feedback controller that can stabilize such systems about any equilibrium does not exist. Thus, suitable nonlinear control techniques have to be resorted to design feedback laws [56, 57]. We chose a dynamic feedback linearization-based controller design that has been treated in detail in [57]. This controller can stabilize the robot about any given configuration in a smooth manner and with exponential convergence.

Uncertainty was added to the robot motion model by adding white noise to the robot dynamics equations aforementioned, with the intensity of the white

noise being approximately 30% of the maximum allowable vehicle linear and angular speed, that is, the noise in the x, y equations had intensity equal to 30% of the maximum allowable linear speed ($\sigma_{x,y} = 0.3v_{\max}$), whereas the noise in the θ equation has intensity equal to 10% of the maximum allowable angular speed ($\sigma_{\theta} = 0.1\omega_{\max}$).

The results of our numerical simulations are shown in Figs. 7 and 8. Figures 7a and 8a represent the tree of feasible trajectories shown without any noise in the system. Figures 7b and 8b show the nominal path, along with the final path ensemble around it. Similarly, Figs. 7c and 8c show the graph built by the GPRM algorithm; however, the edges between the nodes on the graph are only virtual, that is, they are not the actual trajectories. Figures 7d and 8d represent the path ensemble around the nominal trajectory. We performed the experiments

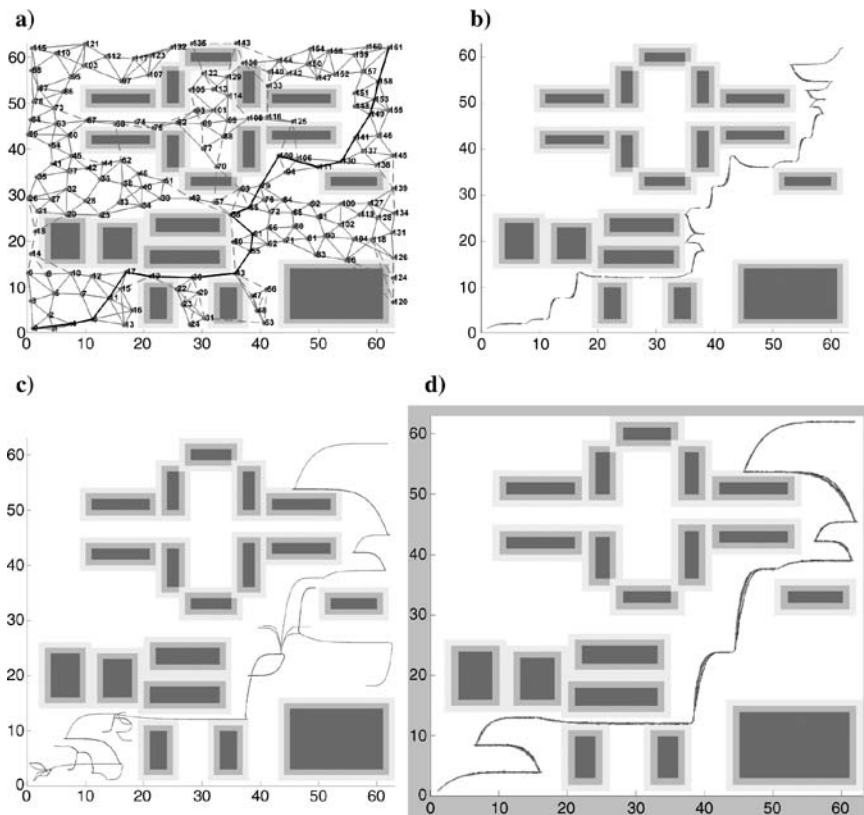


FIG. 7 Performance of GRRT and GPRM on the unicycle robot: map 1 ($p_s \equiv$ probability of success): a) GPRM, b) bundle of final trajectories with $p_s = 100\%$; c) GRRT with nonholonomic constraints, and d) bundle of final trajectories with $p_s = 93.33\%$.

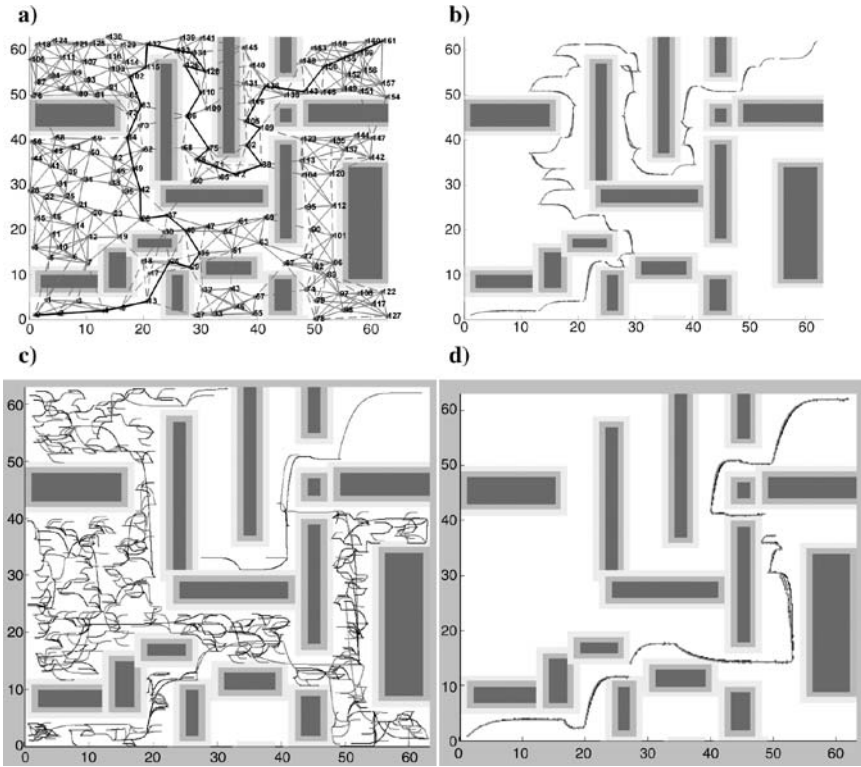


FIG. 8 Performance of GRRT and GPRM on the unicycle robot: map 6 ($p_s \equiv$ probability of success): a) GPRM, b) bundle of final trajectories with $p_s = 100\%$, c) GRRT with nonholonomic constraints, and d) bundle of final trajectories with $p_s = 100\%$.

on the set of six maps that we used in the case of the fully actuated robot, except in this case, because of the nonholonomic constraints on the motion of the robot, the trajectories of the robot are smoother than that in the case of the fully actuated robot. This result is not so surprising because the algorithms are exactly the same for the two cases, except in the design of the local feedback controllers.

GPRM and GRRT are both successful at handling motion uncertainty; however, the GRRT algorithm is easier to search because of the tree structure. One avenue is to use the GRRT algorithm as the option in the GPRM algorithm for large degree-of-freedom (DOF) systems and complicated maps because this approach would control the number of nodes in the GPRM, thereby reducing the search complexity in GPRM while making the options more powerful than if we were to use only primitive local feedback controllers. This case will be one of our future avenues for research. Thus, in this section we have shown the

application of the generalized sampling-based feedback motion planners to both fully actuated and underactuated robotic systems. As shown in the results, the planner has excellent performance in either case in quite complicated maps in the presence of motion uncertainty and uncertainty in the map.

C. CONCLUSION

This section has presented generalized versions of sampling-based motion planners PRM and RRT, that is, GPRM and GRRT. These algorithms generalize the PRM and RRT methodologies to the case where there is uncertainty in both the robot motion model and the map provided to the robot. We have analyzed the algorithms and shown their probabilistic completeness. The algorithms were tested on an idealized planar fully actuated holonomic robot and an under actuated nonholonomic unicycle, and the results clearly show that the performance of the generalized planners is significantly better than the performance of their traditional counterparts, mainly because the traditional PRM and RRT algorithms were not designed to take uncertainty into account.

Although we have obtained very promising results, these algorithms should be applied to robots that can be described by rigid-body equations of motion or any other fully actuated robot system because the systems are essentially feedback equivalent to the planar robot described in this section [58]. Furthermore, resampling using the uniform sampling technique is not the best possible strategy and the need for more sophisticated sampling strategies to increase the efficiency of the planner and increase its probability of success is required. This work was extended in order to handle the motion-planning problem in the presence of motion and map uncertainty for models with rigid-body equations, nonlinear dynamics, and very high-dimensional state-space systems such as n -link manipulators (i.e., up to 16-dimensional state-space problems) with a novel adaptive sampling strategy, which was presented in [59].

Finally, and perhaps most importantly, we would like to relax the assumption that the state of the robot is perfectly known, instead assuming that we only have noisy measurements of the state relative to the map. This approach implies that we need to solve the planning problem in conjunction with the sensing uncertainty, which is a POMDP and is orders of magnitude more complex compared to the MDPs considered in this section. The following section deals with such a problem.

V. SAMPLING-BASED FEEDBACK MOTION PLANNERS WITH MOTION UNCERTAINTY AND IMPERFECT STATE INFORMATION

A. INTRODUCTION AND METHOD OVERVIEW

In robotic systems, decision making under uncertainty is a crucial ability in performing many tasks. Of main sources of uncertainty in robotics systems are uncertainty in determining robot's motion and uncertainty in sensory readings. Under

these uncertainties, a state estimation module can provide a probability distribution over the possible states of the system, and therefore decision making is performed in the space of these distributions, so-called belief space. Planning in the belief space in its most general form is formulated as a POMDP problem [47–49]. However, a very small class of problem, formulated using POMDP, can be solved exactly due to its computational complexity [60, 61]. In particular, planning over continuous state, control, and observation spaces is a big challenge, and in the existing methods these spaces almost always assumed to be discrete.

However, in the absence of uncertainty, sampling-based path planning algorithms such as PRM [62, 63], RRT [64], and their variants have shown a great success in solving robot motion-planning problems. Nevertheless, direct transformation of these methods to planning under uncertainty is a challenge for two main reasons. The first issue is ensuring that the roadmap nodes are reachable. The second challenge is that the incurred costs on different edges of the roadmap depend on each other; this violates a basic assumption in roadmap-based methods that each roadmap edge has to represent an independent planning problem.

In this section, we generalize the PRM framework to the FIRM that takes into account both motion and sensing uncertainties. The probability density function (pdf) over the state is called *belief* or *information state*. The FIRM is constructed as a roadmap in belief space, where its nodes are small subsets of belief space and the edges of FIRM are Markov chains in belief space. FIRM is a generic framework that relies on specific assumptions on its nodes and edges. We construct a LQG-based instantiation of this generic framework, called LQG-based FIRM, which is the first method that generalizes the PRM to belief space in such a way that the incurred costs on different edges of roadmap are independent of each other, while still providing a straightforward approach to sample reachable nodes in belief space. These properties are a direct consequence of utilizing feedback controllers in the construction of FIRM. An important contribution of FIRM framework is that it breaks the curse of history in POMDPs [50] and provides the optimal policy over the whole roadmap instead of only a nominal path.

Figure 9 illustrates the problem of edge dependence in the direct usage of PRM in stochastic domains. Also, it shows the approach of FIRM in handling this problem. Consider a simple PRM with eight nodes $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_8\}$ in Fig. 9a. Assuming the probability distribution over the system state (belief) is Gaussian, we represent belief using a mean \hat{x}^+ and a covariance P , that is, belief is characterized by the pair $b \equiv (\hat{x}^+, P)$. In Fig. 9, covariance is shown by its corresponding 3σ ellipse centered at the mean, which is shown by a small circle. In Fig. 9b, we start with four different beliefs at nodes \mathbf{v}_1 to \mathbf{v}_4 , and we propagate the uncertainty from left to right toward node \mathbf{v}_8 . Although there exists a single edge $e_{(7,8)}$ between nodes \mathbf{v}_7 and \mathbf{v}_8 in PRM (cf. Fig. 9a), the belief evolution along $e_{(7,8)}$ is not unique (cf. Figs. 9b and 9c), and it depends on the initial belief, obtained observation along the path, and the taken path, which has led to \mathbf{v}_7 . Figures 9b and 9c show

this belief propagation in the belief space, where each rectangle encodes a mean and covariance, and as it is seen the belief paths do not form a roadmap in a sense that they do not go to the same beliefs on the PRM nodes.

Moreover, even if we assume there is only one path that leads to ν_7 , every time the robot traverses this path, due to the randomness in measurements, the belief will end up at a different value on ν_7 , and therefore the belief-dependent cost on edge $e_{(7,8)}$ is not predictable without having the full knowledge of the belief at ν_7 or equivalently the full knowledge of history of observations and actions before reaching ν_7 .

In FIRM, using local controllers and appropriate stopping conditions, we stop the belief evolution at predefined unique beliefs associated with PRM nodes (cf. Fig. 9d). Doing so, we break the curse of history by making the after-node belief evolution independent of before-node belief evolution. Thus, we can construct the PRM-like roadmap in belief space, that is, FIRM, with independent edge costs. Thus, once the FIRM is constructed it can be reused for queries from different initial beliefs, and despite the main body of literature in motion planning under uncertainty, FIRM does not need to reproduce all computation every time a new query is submitted.

Besides, without edge independence, the incorporation of obstacles in planning on roadmaps is a challenge because it either entails costly repeated computations of collision probabilities, or makes some simplifying assumptions, such as assuming that the collision probabilities at different stages along a path are independent of each other. Such assumptions in general are not true and may lead to overly conservative plans. However, in FIRM, owing to the independence of edges, we can compute the collision probabilities off-line without such assumptions, and thus we can incorporate obstacles in planning over FIRM that leads to more reliable and less conservative plans.

In the next subsection, we review the most relevant related work. Section V.A provides an overview of the method and its contributions. In Sec. V.B, we discuss the underlying assumptions in FIRM framework, and we derive FIRM MDP as a computationally tractable approximation of POMDP, and we show that FIRM MDP arbitrarily well approximates the original POMDP on a subset of belief space. In Sec. V.C, we present an LQG-based version of generic FIRM framework and show how the underlying assumptions in the FIRM can be satisfied. Section V.D aims at evaluating the quality of the FIRM solution, where we start by extending the concepts of success and probabilistic completeness to the stochastic domains and then prove the probabilistic completeness of the FIRM framework. Experimental results are presented in Sec. V.F and Section V.G concludes the section.

The FIRM graph is a generalization of the PRM graph, whose nodes are small subsets of belief space and whose edges are Markov chains induced by feedback controllers. As a result, planning on FIRM is a MDP, referred to as FIRM MDP here. FIRM MDP is defined on FIRM nodes, and thus it can be solved using standard DP techniques [10].

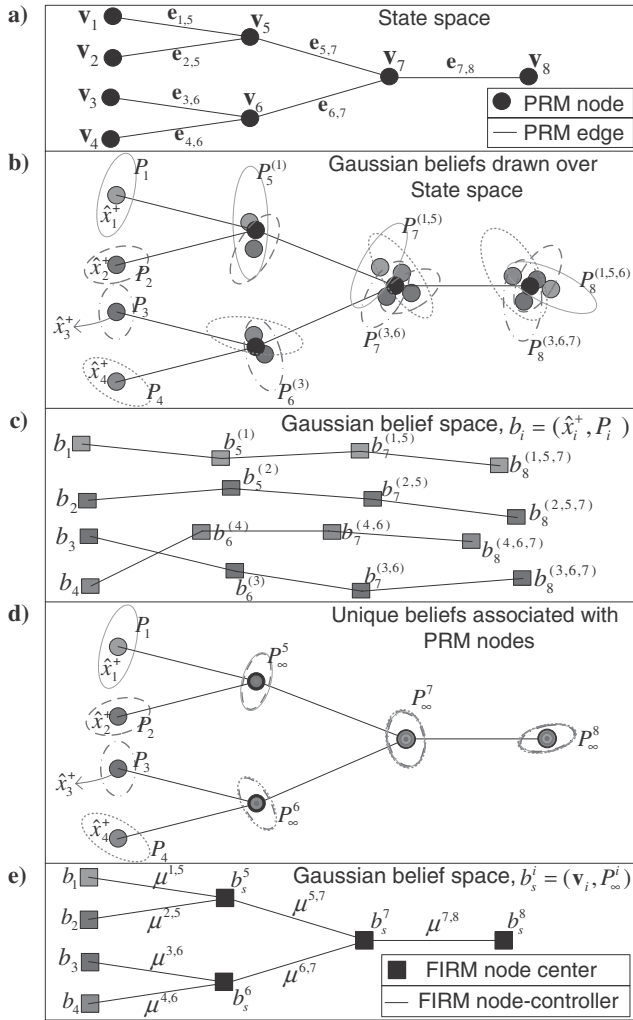


FIG. 9 Problem of edge dependence in the direct usage of PRM in stochastic domains: a) a simple PRM in state space; b) The belief evolution along different paths that leads to the v_8 . As it is seen, the belief depends on the traveled path by robot; for example, $P_8^{(3,6,7)}$ denotes the estimation covariance at node v_8 when the robot has traversed a path through nodes (3,6,7) to reach node 8; c) corresponding belief paths in the belief space where belief at each node depends on the initial belief and history of information; d) unique beliefs assigned to each PRM node, and regardless of the belief history, the belief at each node stops at these predefined beliefs; and e) the FIRM corresponding to the given PRM. Note that μ^j are not FIRM edges. Indeed, (i, j) th FIRM edge is generated by the action of μ^j on b_s^i .

Inducing Reachable Belief Nodes

FIRM samples nodes in the robot state space and then utilizes feedback controllers to automatically induce unique beliefs associated with each of these state-space nodes (cf. Fig. 9). The controller can drive the belief into the neighborhood of these belief states in finite time and thus ensures reachability. This way FIRM addresses the hard task of sampling in reachable belief space that is usually required in belief space planning [40, 65, 66].

Breaking the Curse of History

A fundamental contribution of FIRM is that the optimal action, at a given node, does not depend on the traversed nodes, actions, and observations prior to this node, that is, it is independent of the history of the information process (cf. Fig. 9). This is a direct consequence of inducing reachable belief nodes using the feedback controllers, which essentially breaks the curse of history in POMDPs. In addition, the sampling-based nature of the method borrowed from PRM allows us to ameliorate the curse of dimensionality.

Efficient Planning

The construction of FIRM is off-line, and thus online planning (and replanning) is feasible. Moreover, in FIRM the optimal feedback policy, instead of the nominal path, is computed off-line. This is done by solving the dynamic programming problem associated with the FIRM MDP on belief nodes induced by the feedback controllers.

Incorporating Obstacles in Planning

In the FIRM framework, the collision probabilities can be computed, which leads to more accurate plans, as opposed to a simplified collision measure, that may lead to conservative plans. Obstacles can be included into the planning, while the dependence of collision probability in different steps is taken into account. The obstacles induce a *failure node* in the FIRM MDP, into which the robot can be absorbed. Further, because of the off-line construction of FIRM, the heavy computational burden of estimating collision probabilities can be done off-line, using particle-based techniques, almost instantaneously.

Probabilistic Completeness

Finally, we generalize the conventional concept of “probabilistic completeness” defined for motion-planning methods in the absence of uncertainty to the concept of “probabilistic completeness under uncertainty,” defined for the planners in the presence of uncertainty. According to this definition, we prove that FIRM is a probabilistically complete algorithm. Also, the procedure used in this

proof provides some tools that can be used in analyzing planning methods under uncertainty.

The following algorithms shows the main steps for off-line construction of FIRM. First, the underlying PRM is constructed. Then, FIRM nodes and edges (local controllers) are constructed. Computing transition probabilities and costs, the FIRM MDP, whose solution π^g is a mapping from FIRM nodes to FIRM edges (local controllers) is constructed. Finally, combining the policy π^g over graph with local controllers, we compute the overall policy π from belief space \mathbb{B} to the control space \mathbb{U} , that is, $\pi: \mathbb{B} \rightarrow \mathbb{U}$, which decides which action (control) has to be taken at every belief. Algorithm 5 shows these steps. This construction is off-line. Thus, online planning or replanning is very efficient and reduces to real-time evaluation of function π .

Algorithm 5. Main Steps in Construction of FIRM (off-line)

1. Construct a PRM with nodes $\mathcal{V} = \{v_j\}$ and edges $\mathcal{E} = \{e_{ij}\}$;
2. Construct FIRM nodes $\mathbb{V} = \{B_j\}$ and edges (local controllers) $\mathbb{M} = \{\mu^{ij}\}$;
3. Construct the FIRM MDP, and solve it to compute the optimal mapping $\pi^g: \mathbb{V} \rightarrow \mathbb{M}$;
4. Construct π by combining π^g and local controllers μ^j .

B. FIRM FRAMEWORK

Mainly, uncertainty in planning originates from the lack of exact knowledge on robot's motion model, robot's sensing model, and environment model, which are referred to as motion uncertainty, sensing uncertainty, and map uncertainty, respectively. In this section, we focus on the motion and sensing uncertainty, but some of the concepts are extendible to the problems with a map uncertainty. The MDP problem and POMDP are the most general formulations, respectively, for the planning problem under motion uncertainty and for the planning problem under both motion and sensing uncertainty.

While in the deterministic setting, we seek an optimal path as the solution of motion-planning problem; in the stochastic setting we seek an optimal feedback (mapping) π as the solution of motion-planning problem. In the case of MDP, π is a mapping from the state space to the control space; in the case of POMDP, π is a mapping from the belief space to the control space (see Fig. 10). In this section, we focus on the more general form of POMDPs, whose formulation is presented in Sec. II.B, and we derive the FIRM, as a special case of generic framework presented in Sec. III. To do so, we need to replace the generic state space \mathbb{S} by the belief space \mathbb{B} . As a result, the generic system state s is replaced by the system belief b . Moreover, the nodes $\mathbb{S}_i \subset \mathbb{S}$ centered at s_c^i are replaced by $B_i \subset \mathbb{B}$ centered at belief b_c^i .

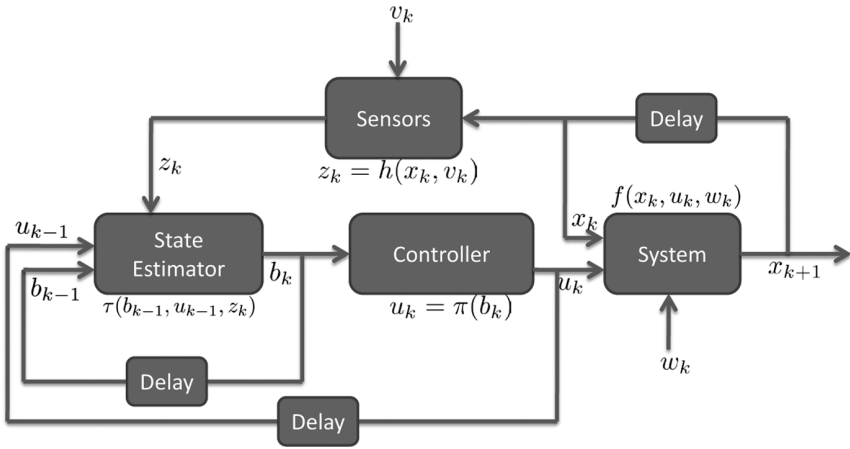


FIG. 10 Block diagram corresponding to the problem of planning under uncertainty.

Therefore, similar to the generic framework, we get the FIRM MDP in the presence of obstacles as

$$J^g(B_i) = \min_{\mathbb{M}(i)} C^g(B_i, \mu^{ij}) + J^g(F) \mathbb{P}^g(F|B_i, \mu^{ij}) + J^g(B_j) \mathbb{P}^g(B_j|B_i, \mu^{ij}) \quad (32a)$$

$$\pi^g(B_i) = \arg \min_{\mathbb{M}(i)} C^g(B_i, \mu^{ij}) + J^g(F) \mathbb{P}^g(F|B_i, \mu^{ij}) + J^g(B_j) \mathbb{P}^g(B_j|B_i, \mu^{ij}) \quad (32b)$$

Again, the overall policy π and initial local controller μ_0 can be computed as follows:

$$\pi : \mathbb{B} \rightarrow \mathbb{U} \quad (33)$$

$$u_k = \pi(b_k) = \begin{cases} \mu_k(b_k), & \mu_k = \pi^g[B(\mu_{k-1})], & \text{if } b_k \in B(\mu_{k-1}) \\ \mu_k(b_k), & \mu_k = \mu_{k-1}, & \text{if } b_k \notin B(\mu_{k-1}) \end{cases}$$

where

$$\mu_0() = \begin{cases} \arg \min_{\mu^{ij} \in \mathbb{M}} C(b_0, \mu^{ij}) + \mathbb{P}(B_j|b_0, \mu^{ij}) J^g(B_j) + \mathbb{P}(F|b_0, \mu^{ij}) J^g(F), & \forall i, j, & \text{if } b \notin \bigcup_m B_m \\ \pi^g(B_l), & & \text{if } \exists l, b \in B_l \end{cases}$$

Again, similarly, the generic algorithms for off-line construction of FIRM and online planning on FIRM are presented in Algorithms 6 and 7, respectively. The concrete instantiations of these algorithms for the LQG-based FIRM are given in the next section.

In summary, in FIRM we aim to solve the original POMDP on a subset of belief space. Because of the condition on the smoothness of cost function and transition probabilities, FIRM solution is arbitrarily close to the solution of original POMDP over FIRM nodes. The important characteristic of FIRM is that it is solved off-line, and thus performing online phase of planning (or replanning) is computationally feasible. To exploit the generic FIRM framework, one has to find a proper (B, μ) pair as the FIRM nodes and edges. Also, there has to be a way of computing transition costs and probabilities. There can be different ways of finding and computing these elements. However, in the next section, we propose a LQG-based FIRM, in which the design of local controllers μ^{ij} and FIRM nodes B_i are based on the properties of LQG controllers. Moreover, we discuss how the transition costs $C^g(B_i, \mu^{ij})$ and the transition probabilities $\mathbb{P}^g(\cdot | B_i, \mu^{ij})$ can be evaluated in LQG-based FIRM. Finally, we solve the corresponding FIRM MDP and provide the algorithms for off-line construction of LQG-based FIRM and online planning with it.

Algorithm 6. Generic Construction of FIRM (Off-line)

1. Construct a PRM with nodes $\mathcal{V} = \{\mathbf{v}_j\}$ and edges $\{\mathbf{e}_j\}$;
2. For each PRM edge \mathbf{e}_j , design a controller μ^{ij} , and compute its corresponding reachable FIRM node B_j ;
3. For each B_i and $\mu^{ij} \in \mathbb{M}(i)$, compute the transition cost $C^g(B_i, \mu^{ij})$, and transition probabilities $\mathbb{P}^g(B_j | B_i, \mu^{ij})$ and $\mathbb{P}^g(F | B_i, \mu^{ij})$ associated with going from B_i to B_j (more rigorously, associated with taking μ^{ij} at B_i);
4. Solve the FIRM MDP (with $\mathbb{S} = \mathbb{B}$) in Eq. (14) to compute feedback π^g over FIRM nodes, and compute the π accordingly.

Algorithm 7. Generic Planning (or Replanning) on FIRM (Online)

1. Given an initial belief b_0 , invoke the controller $\mu_0()$ in Eq. (34) to absorb the robot into some FIRM node B_i ;
2. Given the system is in set B_i , invoke the higher-level feedback policy π^g to choose the lower-level local feedback controller $\mu^{ij}() = \pi^g(B_i)$;
3. Let the local controller $\mu^{ij}()$ execute until absorption into the B_j or failure;
4. Repeat steps 2 and 3 until absorption into the goal node B_{goal} or failure.

C. LQG-BASED FIRM CONSTRUCTION

In this section, we construct a sample FIRM in which local controllers are LQG controllers. We first present a brief review of LQG controllers and then address

how the four elements in the FIRM, that is, nodes B_i , local controllers μ^{ij} , transition probabilities $\mathbb{P}^g(\cdot | B_i, \mu^{ij})$, and costs $C^g(B_i, \mu^{ij})$, can be constructed such that the necessary assumptions in Sec. V.B are satisfied.

Gaussian belief space: Let us denote the estimation vector by x^+ , whose distribution is $b_k = p(x_k^+) = p(x_k | z_{0:k})$. Denote the mean and covariance of x^+ by $\hat{x}^+ = \mathbb{E}[x^+]$ and $P = \mathbb{E}[(x^+ - \hat{x}^+)(x^+ - \hat{x}^+)^T]$, respectively. Denoting the Gaussian belief space by \mathbb{GB} , every function $b(\cdot) \in \mathbb{GB}$ can be characterized by a mean-covariance pair (\hat{x}^+, P) . Abusing the notation, we also show this pair by $b = (\hat{x}^+, P) \in \mathbb{R}^n \times \mathbb{S}_+^n$, where \mathbb{R}^n is the n -dimensional Euclidean space, in which the mean vector \hat{x}^+ lies, and \mathbb{S}_+^n is the space of all positive semi-definite $n \times n$ matrices, in which the covariance matrix P lies.

1. PROPER LQG CONTROLLER

LQG Controllers

The LQG controller is composed of a Kalman filter as the state estimator and LQR controller (see Fig. 10). Thus, the belief dynamics $b_{k+1} = \tau(b_k, u_k, z_{k+1})$ is a known dynamics that comes from the Kalman filtering equations, and the controller $u_k = \mu(b_k)$ that acts on the belief comes from the LQR equations. LQG is an optimal controller for linear systems with Gaussian noises [10]. However, it is also often used for stabilization of nonlinear systems around a given trajectory or around a given point.

Stationary and Time-Varying LQG

Stationary LQG is a time-invariant policy, in which LQG is designed around a given point, say \mathbf{v} , to steer the state of the system to \mathbf{v} . Time-varying LQG is designed to track a given trajectory, in which at every time step a different feedback policy is utilized [10].

From control theory, we know the following lemma is true for stationary LQG [67]. However, in [68] we discussed the proof in detail.

Lemma 1

Consider a stationary LQG controller designed for a controllable and observable linear system with Gaussian noise to regulate the system to point $\mathbf{v} \in \mathbb{X}$. In the absence of stopping region, if we let the controller run and execute the controls, the belief converges in distribution to a random belief, which is denoted by $b_\infty := \lim_{k \rightarrow \infty} b_k$.

Now, we state a result on the properness of stationary LQG controller. But first, we prove the following result.

Lemma 2

If there exists an infinite subsequence of natural numbers $\Xi = \{n_1, n_2, \dots\}$, such that $\mathbb{P}_{n_i}(B|b, \mu) \geq \alpha > 0$ for all i , then (B, μ) pair is a proper pair starting from belief b .

Proof: Based on the definition of \mathbb{P}_{n_i} , the assumption $\mathbb{P}_{n_i}(B|b, \mu) \geq \alpha > 0$ can be rewritten as $\mathbb{P}_{n_i}(b_{n_i} \in B | b_{0:n_i-1} \notin B, b_0 = b, \mu) \geq \alpha > 0$. Denoting the complement of set B by B^c , we can write:

$$\begin{aligned}
 1 - \mathbb{P}(B|b, \mu) &= \mathbb{P}(b_{0:\infty} \notin B | b_0, \mu) = \prod_{k=1}^{\infty} \mathbb{P}(b_{k+1} \in B^c | b_{0:k} \in B^c, b_0, \mu) \\
 &= \prod_{k=1}^{\infty} [1 - \mathbb{P}(b_{k+1} \in B | b_{0:k} \in B^c, b_0, \mu)] \\
 &= \prod_{i=1}^{\infty} [1 - \mathbb{P}(b_{n_i+1} \in B | b_{0:n_i} \in B^c, b_0, \mu)] \\
 &\quad \times \prod_{\substack{k=1 \\ k \notin \Xi}}^{\infty} [1 - \mathbb{P}(b_{k+1} \in B | b_{0:k} \in B^c, b_0, \mu)] \\
 &\leq \prod_{i=1}^{\infty} (1 - \alpha) \prod_{\substack{k=1 \\ k \notin \Xi}}^{\infty} (1) = 0
 \end{aligned} \tag{34}$$

Thus, $\mathbb{P}(B|b, \mu) = 1$, which means that the pair (B, μ) is a proper pair, starting from b . \square

Lemma 3

Consider a stationary LQG controller designed for a linear system with Gaussian noise to regulate the system to point $\mathbf{v} \in \mathbb{X}$ and consider B as its only stopping region. If B is a ball with arbitrary radius $\epsilon > 0$ centered at $\mathbb{E}[b_{\infty}]$, then the pair (B, μ) is proper over the whole Gaussian belief space $\mathbb{G}\mathbb{B}$.

Proof: Let us consider the state-space model of the linear system of interest as follows:

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k + \mathbf{G}w_k, \quad w_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{35a}$$

$$z_k = \mathbf{H}x_k + v_k, \quad v_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{35b}$$

The estimator used in LQG is Kalman filter, and thus the belief update equations, actually, encapsulates the Kalman filtering equations. Thus, the estimation covariance is a deterministic matrix and is computed based on

$$P_k = [I - P_k^- \mathbf{H}^T (\mathbf{H} P_k^- \mathbf{H}^T + \mathbf{R})^{-1} \mathbf{H}] P_k^- \tag{36}$$

where the evolution of P_k^- is governed by the following Riccati equation as follows:

$$P_{k+1}^- = \mathbf{G} \mathbf{Q} \mathbf{G}^T + \mathbf{A} [P_k^- - P_k^- \mathbf{H}^T (\mathbf{H} P_k^- \mathbf{H}^T + \mathbf{R})^{-1} \mathbf{H} P_k^-] \mathbf{A}^T \tag{37}$$

Now, if the P_k^- tends to a stationary covariance, P_k does so. The condition required for P_k^- to converge to a stationary matrix is that the pair (\mathbf{A}, \mathbf{H}) is observable and that the matrix $\mathbf{G}\mathbf{Q}\mathbf{G}^T$ can be decomposed as $\mathbf{M}\mathbf{M}^T$ such that (\mathbf{A}, \mathbf{M}) is controllable [10]. Therefore, for any $\epsilon > 0$, after a deterministic finite time P_k enters the ϵ neighborhood of the stationary covariance, denoted by P_∞ .

However, the dynamics of the estimation mean is not deterministic. If we denote the estimation mean by \hat{x}^+ , in Kalman filtering the dynamics of the estimation mean for the system in Eq. (35) is as follows:

$$\hat{x}_{k+1}^+ = (\mathbf{A} - \mathbf{B}\mathbf{L} - \mathbf{K}_{k+1}\mathbf{H}\mathbf{A})\hat{x}_k^+ + \mathbf{K}_{k+1}\mathbf{H}\mathbf{A}x_k + \mathbf{K}_{k+1}\mathbf{H}\mathbf{G}w_k + \mathbf{K}_{k+1}v_{k+1} \quad (38)$$

where the Kalman gain \mathbf{K}_k is

$$\mathbf{K}_k = P_k^- \mathbf{H}^T (\mathbf{H}P_k^- \mathbf{H}^T + \mathbf{R})^{-1} \quad (39)$$

Thus, if the distribution of observation noise v_{k+1} is Gaussian, for a full rank \mathbf{K}_{k+1} , we have $\mathbb{P}(\hat{x}_{k+1}^+ \in B_{\hat{x}^+} | \hat{x}_{0,k}^+ \notin B, b_0, \mu) > 0$, where $B_{\hat{x}^+}$ is the projection of stopping region B onto the space of possible estimation means. If the $\text{rank}(\mathbf{H}) = \dim(x)$, then \mathbf{K} is full rank for all k , and thus the probability that the mean of estimation eventually enters the its stopping region $B_{\hat{x}^+}$ in finite time is one, that is, we have $\mathbb{P}(\hat{x}^+ \in B_{\hat{x}} | b_0, \mu) = 1$.

Combining the results for estimation covariance and estimation mean, we conclude that the belief $b_k = (\hat{x}_k^+, P_k)$ enters region B in finite time with probability one, based on Lemma 2. Thus, the pair (B, μ) is a proper pair over whole $\mathbb{G}\mathbb{B}$. \square

2. LQG-BASED FIRM NODES B_i AND LOCAL CONTROLLERS μ^U

Underlying PRM

As mentioned, to construct a FIRM, we first construct its underlying PRM. PRM samples its nodes $\{v_j\}_{j=1}^{N_v}$ from \mathbb{X}_{free} based on some appropriate probabilistic measure [62]. Now, we look at the state-space model around every PRM node. In the following, we restrict our approach to the linear models and nonlinear models that are locally well approximated by the linearization. We also assume that both process and measurement noises are drawn from zero-mean Gaussian distributions. Suppose the system (linearized at \mathbf{n}_j) has the state-space form:

$$x_{k+1} = \mathbf{A}^j x_k + \mathbf{B}^j u_k + \mathbf{G}^j w_k, \quad w_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^j) \quad (40a)$$

$$z_k = \mathbf{H}^j x_k + v_k, \quad v_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}^j) \quad (40b)$$

where w_k and v_k are motion and measurement noises, respectively, drawn from zero-mean Gaussian distributions with covariances \mathbf{Q}^j and \mathbf{R}^j .

FIRM Nodes

To design the j th FIRM node, first we design the stationary LQG controller μ_s^j corresponding to the system in Eq. (40) associated with the j th PRM node. The controller μ_s^j is called the j th node controller. Based on Lemma 1, if the system in Eq. (40) is controllable and observable, the $b_\infty^j = (\hat{x}_\infty^j, P_\infty^j)$ exists. Actually, under the

stationary LQG the dynamics of the estimation covariance P_k is deterministic, and it converges to the deterministic covariance P_∞^j [10]. Covariance P_∞^j is computed as $P_\infty^j = (I - L^j H^j) P_\infty^{j-}$, where P_∞^{j-} is the solution following discrete algebraic Riccati equation (DARE) within the class of positive semidefinite symmetric matrices.

$$P_\infty^{j-} = G^j Q^j G^{jT} + A^j [P_\infty^{j-} - P_\infty^{j-} H^{jT} (H^j P_\infty^{j-} H^{jT} + R^j)^{-1} H^j P_\infty^{j-}] A^{jT} \quad (41)$$

$$L^j = P_\infty^{j-} H^{jT} (H^j P_\infty^{j-} H^{jT} + R^j)^{-1} \quad (42)$$

The dynamics of estimation mean \hat{x}_k^+ is random, and it converges to a stationary random vector \hat{x}_∞^{+j} . It is shown in [68] that its mean is $E[\hat{x}_\infty^{+j}] = v_j$. Thus, we can characterize the j th node center:

$$b_s^j = (v_j, P_\infty^j) \quad (43)$$

As a result, considering B_j as a ball with an arbitrary radius $\epsilon > 0$ centered at b_s^j , the pair (B_j, μ_s^j) is a proper pair, based on analysis in Sec. V.C. Thus, one can define B_j as

$$B_j = \{b = (x, P) \mid \|x - v_j\| < \delta_1, \|P - P_\infty^j\| < \delta_2\} \subset \mathbb{R}^n \times \mathbb{S}_+^n \quad (44)$$

where δ_1 and δ_2 are suitably small thresholds that determine the size of FIRM node B_j .

Local Controller

Based on analysis in Sec. V.C, one can choose $\mu^{ij} = \mu_s^j$, as has been done in [69], and due to the shown properness of the stationary LQG controller, the pair (B_j, μ^{ij}) is proper. However, to construct a the local controller μ^{ij} , we precede the node-controller, with a time-varying LQG controller μ_k^{ij} , which is called edge-controller here. Edge-controller μ_k^{ij} is designed by linearizing the system along the (i, j) th PRM edge e_{ij} . The edge-controller has two major roles: first, it exploits the available information on the PRM edges, and thus can buy some safety, that is, distance from obstacles, beforehand. Also, in the case that PRM nodes are not close to each other, it takes the belief into the valid linearization region of the system linearized at j th node, where it hands over the system to the node-controller, and node-controller in turn takes the system to the FIRM node. Thus, the (i, j) th local controller is a concatenation of the (i, j) th edge-controller and j th node-controller.

3. TRANSITION PROBABILITIES AND COSTS

Computing transition probabilities $\mathbb{P}(\cdot | B_i, \mu^{ij})$ and costs $C(B_i, \mu^{ij})$ associated with invoking local controller μ^{ij} at node B_i in general can be a computationally expensive tasks. Here, we utilize the particle-based methods to approximate the distributions and thus compute the collision probabilities. The dependency of collision events in different time steps, which is ignored in most collision probability

computing methods in the POMDP literature, can be taken into account rigorously in MC-based methods. An MC-based approximation can reach any desired accuracy by increasing the number of particles M . However, the main problem of MC-based methods is their high computational cost, which might preclude their use in online scenarios. Nevertheless, owing to the off-line construction of FIRM, the high computational burden of MC-based approaches can be tolerated. The method is detailed in [68].

Depending on the application, one can define a variety of cost functions for taking local controller μ^{ij} at B_i . Here, we first consider estimation accuracy to find the paths, on which the estimator and accordingly controller can perform better. A measure of estimation error is the trace of estimation covariance. Thus, we use

$$\Phi^{ij} = \mathbb{E} \left[\sum_{k=1}^{\mathcal{T}} \text{tr}(P_k^{ij}) \right].$$

In stationary LQG, the covariance matrix evolves deterministically, and thus the expectation operator can be omitted. However, if the filter of choice in edge controller is the extended Kalman filter (EKF), the covariance matrix evolution is also stochastic, and this measure can take into account its stochasticity. Moreover, as we are also interested in faster paths, we take into account the corresponding mean stopping time, that is, $\hat{\mathcal{T}}^{ij} = \mathbb{E}[\mathcal{T}^{ij}]$, and the total cost of invoking μ^{ij} at B_i is considered as a linear combination of estimation accuracy and expected stopping time, with suitable coefficients α_1 and α_2 .

$$C(B_i, \mu^{ij}) = \alpha_1 \Phi^{ij} + \alpha_2 \hat{\mathcal{T}}^{ij} \quad (45)$$

4. OFF-LINE CONSTRUCTION OF LQG-BASED FIRM

The crucial feature of FIRM is that it can be constructed off-line and stored, independent of future queries. Moreover, owing to the reduction from the original POMDP to an N_v -state MDP on belief nodes, the FIRM MDP can be solved using standard DP techniques such as value/policy iteration to yield the optimal policy on graph π^g on the FIRM MDP defined on the belief nodes, which, in turn, at each belief B_i decides which local controller $\mu^* = \pi^g(B_i)$ has to be invoked among all controller $\mu \in \mathbb{M}(i)$. Algorithm 8 details the construction of FIRM.

Algorithm 8. Off-Line Construction of LQG-Based FIRM

1. **input:** Free-space map, \mathbb{X}_{free}
2. **output:** FIRM graph \mathcal{G}
3. Sample PRM nodes $\mathcal{V} = \{\mathbf{v}_j\}_{j=1}^{N_v}$ and construct its edges $\mathcal{E} = \{e_{ij}\}$;
4. **for all** PRM nodes $\mathbf{v}_j \in \mathcal{V}$ **do**
5. Design the node-controller (stationary LQG) μ_s^j about the node \mathbf{v}_j ;

6. Compute associated b_s^j using Eq. (43);
7. Construct FIRM node B_j using Eq. (44);
8. Construct $\mathbb{V} = \{B_i\}$;
9. **for all** PRM edges $e_{ij} \in \mathcal{E}$ **do**
10. Design the edge-controller (time-varying LQG) μ_t^{ij} along the edge e_{ij} ;
11. Construct the local controller μ^j by concatenating edge-controller μ_t^{ij} and node controller μ_s^j ;
12. Set $b_0 = b_s^i$;
13. Generate sample belief paths $b_{0:\mathcal{T}}$ and ground truth paths $x_{0:\mathcal{T}}$ induced by controller μ^{ij} invoked at B_i ;
14. Compute the transition probabilities $\mathbb{P}^g(F|B_i, \mu^{ij})$ and $\mathbb{P}^g(B_j|B_i, \mu^{ij})$ and transition cost $C^g(B_i, \mu^{ij})$;
15. Construct $\mathbb{M} = \{\mu^{ij}\}$;
16. Compute cost-to-go J_g and feedback π^g over the FIRM by solving the DP in Eq. (32);
17. $\mathcal{G} = (\mathbb{V}, \mathbb{M}, J^g, \pi^g)$;
18. **return** \mathcal{G} .

5. PLANNING WITH LQG-BASED FIRM

Given that the FIRM graph is computed off-line, the online phase of planning (and replanning) on the roadmap becomes very efficient and thus feasible in real time. If the given initial belief b_0 does not belong to any B_i , we create a singleton set $B_0 = b_0$. To connect the B_0 to FIRM, we first compute the expected value of the robot state, that is, $\mathbb{E}[x_0]$ using its distribution b_0 and add the $\mathbb{E}[x_0]$ to the PRM nodes, and connect it to the PRM graph. The set of newly added edges going from $\mathbb{E}[x_0]$ to the nodes on PRM is called $E(0)$. We design the local controllers associated with each edge in $E(0)$ and call the set of them $\mathbb{M}(0)$. Then, choosing a local controller in $\mathbb{M}(0)$, the belief enters one of FIRM nodes if no collision occurs. Thus, given the current node, we use policy π^g defined in Eq. (32b) over FIRM nodes to find μ^* and pick μ^* to move the robot into $B(\mu)$. Algorithm 9 illustrates this procedure.

Algorithm 9. Online Phase Algorithm

1. **input:** Initial belief b_0 , FIRM graph \mathcal{G}
2. **If** $\exists B_m \in \mathbb{V}$ such that $b_0 \in B_m$ **then**
3. Set $i = m$, and compute $\mu^* = \pi^g(B_m)$;
4. **else**

5. Compute $\mathbb{E}[x_0]$ based on b_0 , and connect $\mathbb{E}[x_0]$ to the PRM. Call the set of newly added edges $\mathcal{E}(0)$;
6. Design local controllers associated with edges in $\mathcal{E}(0)$. Call the set of these local controllers $\mathbb{M}(0)$;
7. **for all** $\mu \in \mathbb{M}(0)$ **do**
8. Generate sample belief paths $b_{0:\mathcal{T}}$ and ground truth paths $x_{0:\mathcal{T}}$ induced by controller μ invoked at b_0 ;
9. Compute the transition probabilities $\mathbb{P}(F|b_0, \mu)$ and $\mathbb{P}[B(\mu)|b_0, \mu]$ and transition costs $C(b_0, \mu)$;
10. Set $i = 0$, and choose the best local controller $\mu^{ij} = \pi(b_0)$ within the set $\mathbb{M}(0)$ using Eq. (34);
11. **while** $B_i \neq B_{\text{goal}}$ **do**
12. **while** $b_k \notin B_j$ and “no collision” **do**
13. Apply the control $u_k = \mu^{ij}(b_k)$ to the system;
14. Get the measurement z_{k+1} from sensors;
15. **if** Collision happens **then return** Collision;
16. Update belief as $b_{k+1} = \tau[b_k, \mu^{ij}(b_k), z_{k+1}]$;
17. Set $B_i = B_j$, then compute $\mu^{ij} = \pi^q(B_i)$.

D. PROBABILISTIC COMPLETENESS UNDER UNCERTAINTY

In this section, we extend the concept of probabilistic completeness in planning algorithms for deterministic systems to the concept of probabilistic completeness in planning algorithms under uncertainty based on [70]. Accordingly, in the next subsection, we discuss the probabilistic completeness of the FIRM-based algorithms. We start by reviewing the definition of success and probabilistic completeness in the deterministic case, and then we extend these definitions to the stochastic case.

1. SUCCESS IN THE DETERMINISTIC CASE

In the deterministic case, such as conventional PRM, the outcome of the planning algorithm is a path. Thus, success is defined for paths: for a given initial and goal point, a successful path is a path connecting the start point to the goal point, which entirely lies in the obstacle-free space.

2. PROBABILISTIC COMPLETENESS IN THE DETERMINISTIC CASE

In the absence of uncertainty, a sampling-based motion-planning algorithm is probabilistically complete if by increasing the number of samples, the probability of finding a successful path, if one exists, asymptotically approaches to one.

3. DIFFERENCE BETWEEN DETERMINISTIC AND PROBABILISTIC CASE

In the presence of uncertainty, success cannot be defined for a path, and it has to be defined for a policy. Indeed, on a given path different policies may result in different success probabilities. Moreover, under uncertainty one can only assign the probability for reaching goal. Thus, to define a success for a policy, we consider a threshold $p_{\min} \in [0,1]$ and decide about success or failure accordingly.

4. SUCCESSFUL POLICY

In the presence of uncertainty, the solution of the planning algorithm is a policy (feedback) within the class of admissible policies. Therefore, success is defined for policies: for a given initial belief b_0 and goal region B_g , successful policy is a policy within the class of admissible policies under which the probability of reaching goal from the given initial point is greater than some predefined threshold p_{\min} . In other words, $\pi \in \Pi$ is successful for a given b_0 if $\mathbb{P}(\text{success}|b_0, \pi) := \mathbb{P}(B_g|b_0, \pi) > p_{\min}$.

5. FEASIBLE INITIAL BELIEF

A belief $b_0 \in \mathbb{B}$ is a feasible initial belief for class Π if there exists a policy $\pi \in \Pi$ such that $\mathbb{P}(\text{success}|b_0, \pi) > p_{\min}$. The set of all feasible initial beliefs corresponding to a class Π is denoted by \mathbb{B}_{Π} .

$$\mathbb{B}_{\Pi} = \{b_0 \in \mathbb{B} : \exists \pi \in \Pi, \mathbb{P}(\text{success}|b_0, \pi) > p_{\min}\} \quad (46)$$

The richer the set of admissible policies Π is, the greater the set of feasible initial beliefs \mathbb{B}_{Π} . For example, in obstacle-free LQG-based FIRM, the set of all Gaussian beliefs $\mathbb{G}\mathbb{B}$ is a subset of \mathbb{B}_{Π} .

6. GLOBALLY SUCCESSFUL POLICY

Instead of a single initial belief, we can also define the concept of successful policy for \mathbb{B}_{Π} . For a given goal region B_g , policy $\pi \in \Pi$ is globally successful if the probability of reaching goal from any belief in \mathbb{B}_{Π} is greater than p_{\min} . In other words, $\pi \in \Pi$ is globally successful if $\mathbb{P}(\text{success}|b_0, \pi) = \mathbb{P}(B_g|b_0, \pi) > p_{\min}, \forall b_0 \in \mathbb{B}_{\Pi}$.

7. PROBABILISTIC COMPLETENESS UNDER UNCERTAINTY

Probabilistic completeness can be defined based on either one of the definitions for the successful policy. Suppose there exists a (globally) successful policy $\pi \in \Pi$. Then, a sampling-based motion-planning algorithm is probabilistically complete under uncertainty, if by increasing the number of samples without bound the probability of finding a (globally) successful policy is one. In other words, for the globally successful case, if there exists a globally successful policy

$\pi \in \Pi$, we have following property:

$$\lim_{N_v \rightarrow \infty} \mathbb{P}[B_g | b_0, \pi(\cdot; \mathcal{V})] > p_{\min}, \quad \forall b_0 \in \mathbb{B}_\Pi \quad (47)$$

where $\mathcal{V} = \{v_i\}_{i=1}^{N_v}$ is the set of nodes of underlying PRM. Notation $\pi(\cdot; \mathcal{V})$ emphasizes the fact that PRM nodes \mathcal{V} appear as parameters in the policy π . If one is not interested in globally successful policy, the definition is the same as Eq. (47); only instead of $\forall b_0 \in \mathbb{B}_\Pi$, one has to consider a given b_0 , from which the policy starts.

E. PROBABILISTIC COMPLETENESS OF FIRM

In this section, we present a result on the probabilistic completeness of the FIRM-based methods. We first provide an analysis of the local planners in belief space and state the necessary assumptions rigorously. Finally, the main theorem and its proof are stated for the more general case of globally successful policies.

1. NOTATION

The norm $\|\cdot\|$ is the supremum norm, when it is applied to functions. The norm $\|\cdot\|_{\text{op}}$ is applied on operators, and it stands for the operator norm [71]. In this section, by the word “continuous” we mean “Lipschitz continuous.”

2. HYPERSTATE

$\mathcal{X} = (x, b) \in \mathbb{X}_h$ is referred to as hyperstate (or h-state), which is the state-belief pair. The space of all h-states is called hyperstate space (h-state space) $\mathbb{X}_h = \mathbb{X} \times \mathbb{B}$. The $p^\mu(\mathcal{X}' | \mathcal{X})$ denotes the one-step transition pdf induced by the local controller μ , over the h-state space. Also, $\mathbb{P}_n(\cdot | \mathcal{X}, \mu) : \mathbb{B}_{\mathbb{X}_h} \rightarrow [0, 1]$ is the probability measure over the h-state space, induced by the local controller μ after n steps, starting from the h-state \mathcal{X} . Set $\mathbb{B}_{\mathbb{X}_h}$ is the sigma-algebra of the h-state space \mathbb{X}_h .

3. LOCAL PLANNER AND EXTENDED STOPPING REGION

The role of the (i, j) th local planner or local controller in the belief space is to drive the belief from the belief node B_i to the belief node B_j in belief space, whose stopping region is B_j . In the presence of obstacles, we extend the concept of stopping region to include the obstacles also. Both the stopping regions in the belief space $\{B_j\}$ and the stopping region in the state space F can be extended to the h-state space, respectively, denoted by $\{\mathcal{B}_j\}$ and \mathcal{F} where $\mathcal{B}_j \subset \mathbb{X}_h$ and $\mathcal{F} \subset \mathbb{X}_h$:

$$\mathcal{B}_j = \{(X, b) | X \in \mathbb{X}_{\text{free}}, b \in B_j\} \quad (48)$$

$$\mathcal{F} = \{(X, b) | X \in F, b \in \mathbb{B}\} \quad (49)$$

$$\mathcal{S}_j = \mathcal{B}_j \cup \mathcal{F}, \quad \overline{\mathcal{S}}_j = \mathbb{X}_h \setminus \mathcal{S}_j \quad (50)$$

where \mathcal{S}_j and $\overline{\mathcal{S}}_j$, respectively, denote the entire stopping region and transient region under the local controller μ^{ij} .

4. ABSORPTION PROBABILITY OF LOCAL PLANNERS

If under the dynamics induced by the local planner, the system reaches the target node \mathcal{B}_j , the local planner is considered to be successful, and if the system hits an obstacle, the local planner is considered to be failed. The success probability of the local planners or the absorption probability into the FIRM nodes are computed through solving following integral equation that comes from the law of total probability:

$$\begin{aligned} \mathbb{P}(\mathcal{B}_j|\mathcal{X}, \mu^{ij}) &= \int_{\mathbb{X}_n} p^{\mu^{ij}}(\mathcal{X}'|\mathcal{X})\mathbb{P}(\mathcal{B}_j|\mathcal{X}', \mu^{ij}) d\mathcal{X}' \\ &= \int_{\mathcal{B}_j} p^{\mu^{ij}}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' + \int_{\overline{\mathcal{S}}_j} p^{\mu^{ij}}(\mathcal{X}'|\mathcal{X})\mathbb{P}(\mathcal{B}_j|\mathcal{X}', \mu^{ij}) d\mathcal{X}' \end{aligned} \quad (51)$$

where the second equality in Eq. (51) follows from substituting the following conditions, inherited from FIRM construction, into the first integral:

$$\mathbb{P}(\mathcal{B}_j|\mathcal{X}, \mu^{ij}) = \begin{cases} 1, & \text{if } \mathcal{X} \in \mathcal{B}_j \\ 0, & \text{if } \mathcal{X} \in \mathcal{F} \end{cases} \quad (52)$$

Henceforth, we drop the indices i and j to unclutter the expressions. Thus, we can write

$$\begin{aligned} \mathbb{P}(\mathcal{B}|\mathcal{X}, \mu) &= \int_{\mathcal{B}} p^{\mu}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' + \int_{\overline{\mathcal{S}}} p^{\mu}(\mathcal{X}'|\mathcal{X})\mathbb{P}(\mathcal{B}|\mathcal{X}', \mu) d\mathcal{X}' \\ &= R(\mathcal{X}) + T_S[\mathbb{P}(\mathcal{B}), \mu](\mathcal{X}) \end{aligned} \quad (53)$$

where the operator T_S and the function $R(\mathcal{X})$ are defined as

$$T_S[f(\cdot)](\mathcal{X}) := \int_{\overline{\mathcal{S}}} p^{\mu}(\mathcal{X}'|\mathcal{X})f(\mathcal{X}') d\mathcal{X}', \quad R(\mathcal{X}) := \int_{\mathcal{B}} p^{\mu}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' \quad (54)$$

The solution of the integral equation in Eq. (53) is expressed in the following as a Liouville–Neumann series [71], similar to the solution of inhomogeneous Fredholm equation of second type [71]:

$$\mathbb{P}(\mathcal{B}|\mathcal{X}, \mu) = \sum_{n=1}^{\infty} T_S^n[R(\cdot)](\mathcal{X}) \quad (55)$$

We show that the series in Eq. (55) is a convergent series, by resorting to the following assumption, which is a weaker version of the aforementioned FIRM condition on the design of node-local controller pair.

Assumption 2: We assume that after a sufficient number of steps, say, $N < \infty$, the probability of arriving into S_j , the stopping region of μ^j , is greater than arbitrary number $\beta > 0$ regardless of the starting point, that is, we assume for each μ^j , $\mathbb{P}_n(S_j | \mathcal{X}, \mu^j) \geq \beta > 0$, for all $n > N$ and \mathcal{X} .

This assumption is a reasonable assumption for a properly designed controller as it rephrases the role of controller, in a probabilistic setting, in driving the system toward the target region. (The condition $\forall n > N$ can be replaced by a weaker statement of “infinitely often,” defined in probability theory [72]). In LQG-based FIRM, this assumption always holds on bounded environments.

Lemma 4

Given Assumption 2, we have

$$\begin{cases} \|\mathbf{T}_S^n\|_{\text{op}} \leq 1, & n < N \\ \|\mathbf{T}_S^n\|_{\text{op}} \leq 1 - \beta < 1, & n \geq N \\ \sum_{n=0}^{\infty} \|\mathbf{T}_{S^{\text{op}}}^n\| \leq c < \infty \end{cases} \quad (56)$$

Proof: See Sec. A in Appendix. \square

Corollary 1: Series $\sum_{n=0}^{\infty} \mathbf{T}_S^n[R]$ is a convergent series, and, therefore, we can define the resolvent operator $(I - \mathbf{T}_S)^{-1}[R] = \sum_{n=0}^{\infty} \mathbf{T}_S^n[R]$, where $\|(I - \mathbf{T}_S)^{-1}\|_{\text{op}} \leq c < \infty$.

Proof: See Sec. B in Appendix. \square

According to the corollary 1, the success probability can be written using the defined resolvent operators as

$$\mathbb{P}(\mathcal{B} | \mathcal{X}, \mu) = (I - \mathbf{T}_S)^{-1}[R(\cdot)](\mathcal{X}) \quad (57)$$

As the first result of this section (proposition 1), we aim to show that this absorption probability varies continuously with respect to changes in the parameters of the local planner. However, we first state two more assumptions.

Assumption 3: We assume the local planning law and induced transition probabilities are smooth, that is,

1. Local control laws are continuous in their parameters, that is, for the j th local controller, mapping $\mu^j(\mathbf{v}_j): \mathbb{B} \rightarrow \mathbb{U}$ is a continuous function in its parameter \mathbf{v}_j .
2. The transition pdf on h-state, that is, $p(\mathcal{X}' | \mathcal{X}, u)$ is a continuous function of control u , that is, there exists a $c_1 < \infty$, such that $\|p(\mathcal{X}' | \mathcal{X}, u) - p(\mathcal{X}' | \mathcal{X}, \tilde{u})\| \leq c_1 \|u - \tilde{u}\|$.

Finally, we state the following assumption, in which we emphasize the fact that as $\mathbf{v} \rightarrow \tilde{\mathbf{v}}$, the probability measure induced by the local controller $\mu(\cdot; \mathbf{v})$ over the sets \mathcal{B} and $\tilde{\mathcal{B}}$ have to converge also, which is a reasonable assumption for a smooth control law.

Assumption 4: Consider the controllers $\mu(\cdot; \mathbf{v})$, and $\check{\mu}(\cdot; \check{\mathbf{v}})$, whose corresponding extended absorption regions are denoted by \mathcal{B} and $\check{\mathcal{B}}$, respectively. We assume that there exist real numbers $r > 0$ and $c' < \infty$, such that for $\|\mathbf{v} - \check{\mathbf{v}}\| \leq r$, we have

$$\|\mathbb{P}_1(\mathcal{B} \ominus \check{\mathcal{B}} | \mathcal{X}, \mu)\| \leq c' \|\mathbf{v} - \check{\mathbf{v}}\| \quad (58)$$

where \ominus is the symmetric difference operator, that is, $\mathcal{B} \ominus \check{\mathcal{B}} = (\mathcal{B} \setminus \check{\mathcal{B}}) \cup (\check{\mathcal{B}} \setminus \mathcal{B})$.

Now, we state following proposition on the continuity of success probability of local planners:

Proposition 1: (Continuity of absorption probabilities): Given assumptions 2, 3, and 4, the absorption probability $\mathbb{P}(\mathcal{B}_j | b, \mu^j)$ is continuous in parameter \mathbf{v}_j , for all i, j , and b .

Proof: See Sec. C in Appendix. □

Now, we present the main result on the probabilistic completeness of the FIRM-based methods:

Theorem 1: Given assumptions 2, 3, and 4, any planning algorithm under uncertainty that is generated based on FIRM framework (i.e., guarantees belief-node reachability and induces a roadmap in the belief space with independent local planners) is probabilistically complete under uncertainty.

Proof: See Sec. D in Appendix. □

The basic idea in probabilistic completeness under uncertainty stems from an idea similar to the one in path isolation-based analysis in planners for deterministic systems. Roughly speaking, in the path isolation argument for the sampling-based planners in the absence of uncertainty, if there is a successful path and a nonzero neighborhood of this path in which every path is successful, we can eventually find a path in this neighborhood, by increasing the number of samples unboundedly. Similarly, in the presence of uncertainty, if there is a successful policy, it is parameterized by some parameters. Thus, if there exists a nonzero measure neighborhood of these parameters, in which every selected parameter leads to a successful policy, we can eventually reach a successful policy by increasing the number of samples unboundedly and falling into the target neighborhood.

F. EXPERIMENTAL RESULTS

In this section, we first illustrate the theoretical results in preceding sections on a simple example experiment in a small three-dimensional planning domain. Then, we present planning results on a larger three-dimensional state space.

1. MOTION MODEL

A three-wheel omnidirectional mobile robot is used in experiments with the nonlinear kinematic model given in [73]. The state vector is composed of a two-dimensional location and heading angle $x = [^1x, ^2x, \theta]^T$ at global world frame. Here $u = [^1u, ^2u, ^3u]^T$ is the vector of controls, where iu is the linear velocity of the i th wheel. Also, w is the motion noise, which is drawn from a zero-mean Gaussian distribution. The motion dynamics for this robot in its original continuous form is [73]:

$$\dot{x} = f_c(x, u, w) = T(x)u + w \quad (59)$$

where

$$T(x) = \begin{pmatrix} -\frac{2}{3}\sin(\theta) & -\frac{2}{3}\sin(\frac{\pi}{3} - \theta) & \frac{2}{3}\sin(\frac{\pi}{3} + \theta) \\ \frac{2}{3}\cos(\theta) & -\frac{2}{3}\cos(\frac{\pi}{3} - \theta) & -\frac{2}{3}\cos(\frac{\pi}{3} + \theta) \\ \frac{1}{3r} & \frac{1}{3r} & \frac{1}{3r} \end{pmatrix} \quad (60)$$

where r is the distance of wheels from the robot's mass center. The discrete motion dynamics is shown by

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \quad (61)$$

$w_k \sim \mathcal{N}(0, Q)$ is the motion noise at k th time step, which is drawn from a zero-mean Gaussian distribution with covariance matrix Q . It can be shown that if we linearize this system the linearized motion model is controllable. Therefore, the stationary LQG and the designed stopping region in Eq. (44) form a proper pair. The linearized model of nonholonomic robots such as the unicycle model is not necessarily controllable around a point, and thus the stationary LQG does not seem to be an appropriate choice in that case. However, as we discuss in future work section, using time-varying controllers we can construct proper (B, μ) pairs and thus construct a FIRM framework for nonholonomic motion models.

2. OBSERVATION MODEL

In experiments, the robot is equipped with exteroceptive sensors that provide range and bearing measurements from existing landmarks (radio beacons) in the environment. The two-dimensional location of the j th landmark is denoted by L_j . Measuring L_j can be modeled as follows:

$$j_z = {}^jh(x, {}^jv) = \left[\|{}^jd\|, \text{atan2}({}^jd_2, {}^jd_1) - \theta \right]^T + {}^jv, \quad {}^jv \sim \mathcal{N}(0, {}^jR)$$

where ${}^jd = [{}^jd_1, {}^jd_2]^T := [^1x, ^2x]^T - L_j$. The vector jv is a state-dependent observation noise, with covariance

$${}^jR = \text{diag}[(\eta_r \|{}^jd\| + \sigma_b^r)^2, (\eta_\theta \|{}^jd\| + \sigma_b^\theta)^2] \quad (62)$$

In other words, the uncertainty (standard deviation) of sensor reading increases as the robot gets farther from the landmarks. Here $\eta_r = \eta_\theta = 0.3$ determines this dependence, and $\sigma_b^r = 0.01$ m and $\sigma_b^\theta = 0.5$ deg are the bias standard deviations. A similar model for range sensing is used in [36]. We assume the robot observes all N_L landmarks at all times, and their observation noises are independent. Thus, the total measurement vector is denoted by $z = [{}^1z^T, {}^2z^T, \dots, {}^{N_L}z^T]^T$, and because of the independence of measurements of different landmarks, an observation model for all landmarks can be written as

$$z = h(x) + v, \quad v \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \quad \mathbf{R} = \text{diag}({}^1\mathbf{R}, \dots, {}^{N_L}\mathbf{R}) \quad (63)$$

3. CONSTRUCTION OF LQG-BASED FIRM NODES AND EDGES

Figure 11a shows a sample environment, including obstacles, landmarks, and enumerated nodes in $({}^1x, {}^2x, \theta)$ space. Nodes are shown by small triangles, which encode the position $({}^1x, {}^2x)$ and heading angle θ of the robot. Drawn stars represent landmarks. The corresponding FIRM nodes are computed and shown in Fig. 11b. All elements in Fig. 11b are defined in $({}^1x, {}^2x, \theta)$ space, but only the $({}^1x, {}^2x)$ portion of them is shown here. Each $b_s^j = (\mathbf{v}_j, P_\infty^j)$ is illustrated by an ellipse, representing 3σ ellipse of covariance P_∞^j , centered at a point representing \mathbf{v}_j . Each FIRM node B_j is a neighborhood around b_s^j . In the experiments, we define the node region using the component-wise version of Eq. (44) to handle the error scale difference in position and orientation variables:

$$B_j = \{b = (x, P) \mid |x - \mathbf{v}_j| < \epsilon, |P - P_\infty^j| < \Delta\} \quad (64)$$

where $||$ and $\dot{<}$ stand for the absolute value and component-wise comparison operators, respectively. We set $\epsilon = [0.07 \text{ (m)}, 0.07 \text{ (m)}, 1 \text{ (deg)}]^T$ and $\Delta = \epsilon \epsilon^T$ to quantify B_j . Projection of B_j onto the space of estimation mean, that is, $B_j^x = \{\hat{x}^+ : |\hat{x}^+ - \mathbf{v}_j| < \epsilon\}$ is a neighborhood around \mathbf{v}_j , which is shown by a small rectangle centered at \mathbf{v}_j . Projection of B_j onto the space of estimation covariances, that is, $B_j^P = \{P : |P - P_\infty^j| \dot{<} \Delta\}$ is a neighborhood around P_∞^j . However, in a two-dimensional plot B_j^P cannot be shown due to its high dimension. Thus, we partially illustrate it only by two dashed ellipses that represent 3σ covariances of $P_\infty^j - \Delta_d$ and $P_\infty^j + \Delta_d$, where Δ_d is the matrix Δ , whose off-diagonal elements are set to zero. For illustration purposes, both of these neighborhoods, that is, B_j^x and B_j^P , are five times magnified in Fig. 11b.

4. TRANSITION COSTS AND PROBABILITIES

After designing FIRM nodes and local controllers, the transition costs and probabilities have to be computed. Based on task and needed accuracy, different approaches can be taken. Here, we use a particle-based approximation of distribution to

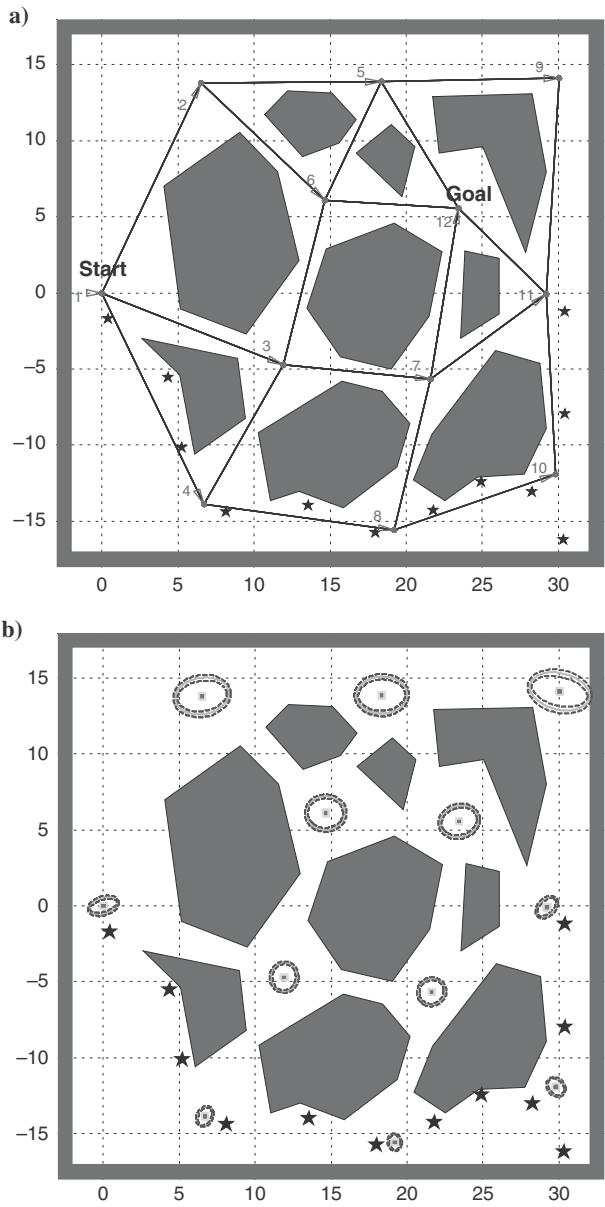


FIG. 11 Sample environment: a) figure depicts the underlying PRM graph where gray polygons are the obstacles, and black stars represent the landmarks' locations; b) FIRM nodes corresponding to PRM nodes.

compute these quantities, and we use $M = 100$ particles. In other words, for every (B, μ) pair, we perform 100 runs. At every run, a sample path of state x , a sample path of estimation mean \hat{x}^+ , and a sample path of estimation covariance P are generated. If the filter of choice in the edge-controller is the linearized Kalman filter (LKF) [74, 75], the covariance evolution is deterministic, and there is no need to generate 100 different sample covariance paths. However, if the filter of choice in the edge-controller is the EKF [74, 75], we have to generate the sample covariance paths too, to take into account the stochasticity of the covariance matrix. Figure 12a depicts the sample paths of the true state x and estimation mean \hat{x}^+ in light and dark colors, respectively, for $M = 100$ particles. Note that when a true state path (light-color path) collides with an obstacle, the process stops, and failure happens. However, in this figure for illustration purposes we continue the process and ignore the obstacles to better show the uncertainty tube and information availability at different parts of the space. As seen in Fig. 12a, the behavior of the true state on the edges, which have access to more accurate observations, is remarkably closer to the planned behavior. In contrast, on the edges that get less informative observations, the controller cannot effectively compensate for the deviations of the ground truth from the nominal path, which can lead to collision with obstacles.

To avoid clutter, Fig. 12b depicts sample estimation covariance evolution only for a single particle. In this figure, we let the process and observation noises be zero to keep the center of ellipses (i.e., estimation mean) on the planned points. However, note that, in general, estimation mean is affected by the noise (as it is seen in Fig. 12a). Indeed, Fig. 12b can be seen as the maximum-likelihood estimation uncertainty tube over the roadmap.

Let us denote the q th sample path for the true state by $x_{0:\mathcal{T}^q}^{(q)}$, for the estimation mean by $\hat{x}_{0:\mathcal{T}^q}^{+(q)}$, and for the estimation covariance by $P_{0:\mathcal{T}^q}^{(q)}$, where \mathcal{T}^q is the stopping time of the q th particle in executing μ at B . Moreover, one can assign a weight to each run q based on the its probability of occurrence. There are different ways proposed to compute these weights in the sequential Monte Carlo literature [76]. However, the main condition is that they have to sum to one, that is, $\sum_{q=1}^M w^{(q)} = 1$. Here we simply consider $w^{(q)} = M^{-1}$. Note that if we run μ^{ij} at B_i , all of these quantities also have to have a ij superscript. Now, having these sample paths, we can compute the transition costs and probabilities associated with invoking the μ^{ij} at B_i . For the collision probability, we have

$$\mathbb{P}^g(F|B_i, \mu^{ij}) = \mathbb{E}[\mathbb{I}_F|B_i, \mu^{ij}] \approx \sum_{q=1}^M w^{(q)} \mathbb{I}_F[x_{0:\mathcal{T}^q}^{(q)}] \quad (65)$$

$$\mathbb{P}^g(B_j|B_i, \mu^{ij}) = 1 - \mathbb{P}^g(F|B_i, \mu^{ij}) \quad (66)$$

where \mathbb{I}_F is the failure indicator. It is one if there exists a time step $k \leq \mathcal{T}^{(q)}$, such that $x_k \in F$. Otherwise it is zero. \mathcal{T}^q , or more rigorously $\mathcal{T}^{ij^{(q)}}$, is the stopping time of q th particle in executing μ^{ij} at B_i . To compute $\mathcal{T}^{ij^{(q)}}$ defined in Eq. (11), we only need to check the condition $b \in B_j$ at every time step and find the first time step

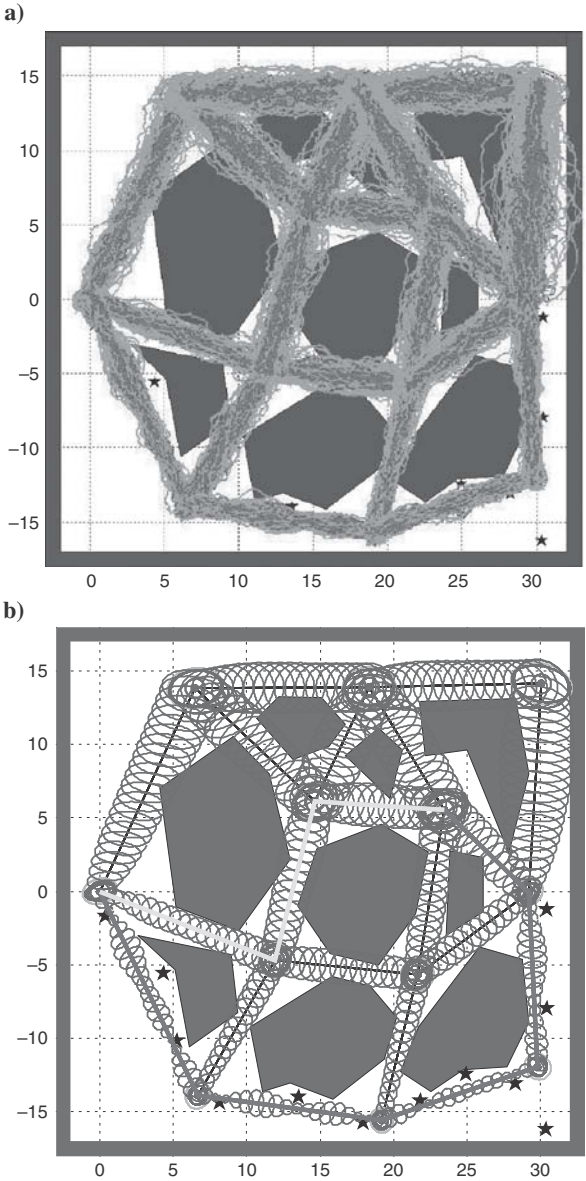


FIG. 12 Sample paths induced by the controllers invoked in different nodes: a) for $M = 100$ particles, the sample ground truth paths and the sample estimation mean paths are shown in light and dark colors, respectively; and b) the most likely path under the optimal policy and shortest path are shown in bold dark and bold paths, respectively, and the 3σ ML estimation uncertainty tube is drawn by a tube of ellipses.

that belief b enters the stopping region B_j . Thus, we can compute the mean stopping time as

$$\hat{\mathcal{T}}^{ij} = \mathbb{E}[\mathcal{T}^{ij}] \approx \sum_{q=1}^M w^{(q)} \mathcal{T}^{ij(q)} \quad (67)$$

To compute the filtering cost defined in Sec. V.C, again we use the particle-based representation of belief:

$$\Phi^{ij} = \mathbb{E} \left[\sum_{k=1}^{\mathcal{T}^{ij}} \text{tr}(P_k) | B_i, \mu^{ij} \right] \approx \sum_{q=1}^M \sum_{k=1}^{\mathcal{T}^q} w^{(q)} \text{tr} P_k^{(q)} \quad (68)$$

where $P_K^{(q)}$ is the estimation covariance at k th time step of q th particle. Finally, the cost of taking μ^{ij} at B_j is as follows:

$$C(B_i, \mu^{ij}) = \alpha_1 \Phi^{ij} + \alpha_2 \hat{\mathcal{T}}^{ij}$$

where we used the coefficients $\alpha_1 = 0.8$ and $\alpha_2 = 0.2$. Table 1 shows these quantities for several (B_i, μ^{ij}) pairs in corresponding to Fig. 12.

5. PLANNING AND REPLANNING ON FIRM

Plugging the computed transition costs and probabilities into the Eq. (32), we can solve the DP and compute the graph policy π^g . This process is performed once off-line. Figure 13a shows the policy π^g on the constructed FIRM in this example. Indeed, at every FIRM node B_i , the policy π^g decides which local controller has to be taken, which in turn aims to take the robot to the next FIRM node. Thus, the online part of planning is significantly efficient and only reduces to executing the controller and generating control signal, which is almost an instantaneous computation.

The great consequence of this framework is that replanning can be performed using FIRM efficiently. Suppose due to some unmodeled large disturbance, the robot's belief deviates significantly from the planned path, that is, for some appropriate norm $\| \cdot \|$ on belief space we have $|b_k - \mathbb{E}[b_k^p]| > \varrho$, where b_k^p is the planned belief at k th time step, and ϱ is the threshold for deciding if replanning is needed or not. In such cases, replanning occurs and is based on Algorithm 9. In Fig. 3b, we illustrate a simple replanning process. In this figure it is assumed that an unmodeled large disturbance affects the system, such that the estimation mean significantly deviates from the planned path. The deviated mean is shown on the figure as the "restart point." Thus, based on Algorithm 9, we connect this point to the PRM. In Fig. 3b the newly added PRM edges, that is, $\mathcal{E}(0)$, is shown by dashed lines. Then, for every edge in $\mathcal{E}(0)$, we design a local controller. Call the set of these newly constructed local controllers $\mathbb{M}(0)$. For every $\mu \in \mathbb{M}(0)$, compute corresponding transition costs and probabilities. Finally, according to Bellman's

TABLE 1 COMPUTED COSTS FOR SEVERAL NODE-CONTROLLER PAIRS IN FIRM USING 100 PARTICLES

(B_i, μ^{ij}) pair	$B_1, \mu^{1,4}$	$B_4, \mu^{4,8}$	$B_8, \mu^{8,10}$	$B_{10}, \mu^{10,11}$	$B_{11}, \mu^{11,12}$	$B_1, \mu^{1,3}$	$B_3, \mu^{3,6}$	$B_6, \mu^{6,12}$
$\mathbb{P}^g(B_j B_i\mu)$	%97	%95	%99	%77	%79	%87	%55	%79
Φ^{ij}	18.5967	11.2393	6.8229	15.1148	26.2942	23.6183	48.8189	43.6207
$\mathbb{E}[\mathcal{T}^{ij}], \sigma[\mathcal{T}^{ij}]$	238.2, 21.8	193.0, 28.7	150.0, 15.1	209.6, 24.5	170.8, 22.6	200.3, 22.7	242.4, 30.1	219.2, 26.7

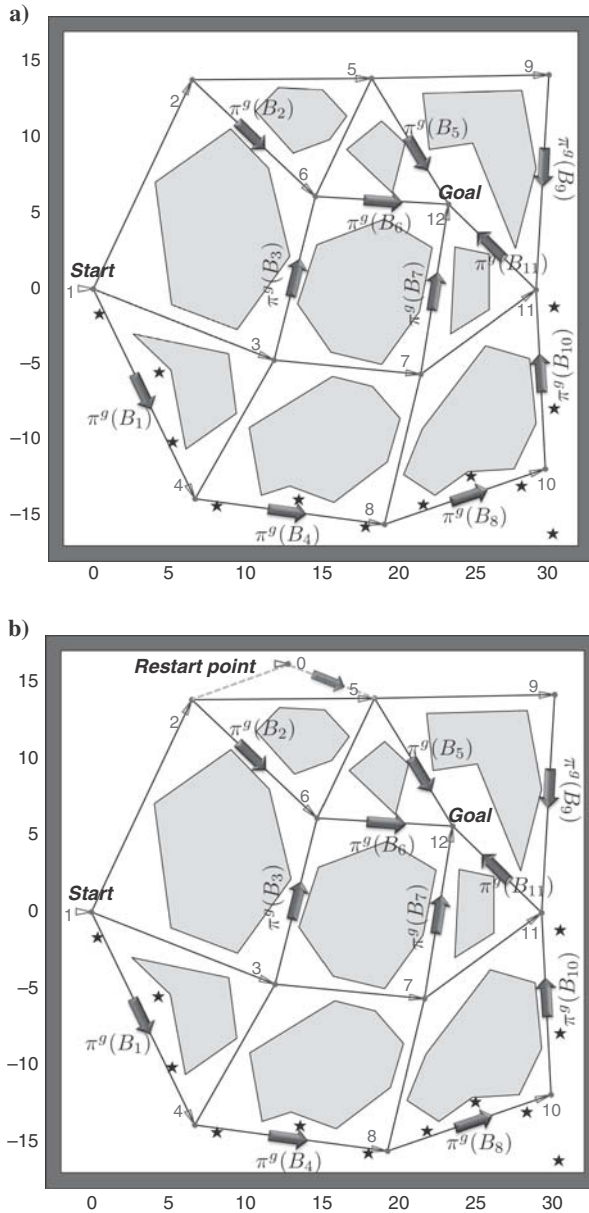


FIG. 13 Planning and replanning on FIRM: a) policy π^g resulted from solving DP in Eq. (32) is shown by large arrows, and indeed for every FIRM node, the policy π^g tells which controller has to be taken; and b) in this figure it is assumed that an unmodeled large disturbance affects the system, such that the estimation mean significantly deviates from the planned path. The deviated mean is shown by α .

principle of optimality, we use the precomputed cost-to-gos $J^g()$ to decide which controller has to be taken at the restart point using Eq. (34). Taking this controller, the belief state returns to the FIRM nodes, and from there again we can use pre-computed π^g to control the robot toward the goal region.

We show the most likely path under the π^g by bold dark lines in Fig. 12b. The shortest path is also illustrated in Fig. 12b by bold light lines. It can be seen that the “most likely path under the best policy” detours from the shortest path to a path along which the filtering uncertainty is smaller, and it is easier for the controller to avoid the collisions.

6. LARGER ENVIRONMENT

In this subsection, we look at a larger environment whose size is 10,000 m². We construct a PRM in \mathbb{X}_{free} , which has 162 nodes and 608 edges. Figure 14 shows this PRM projected to two-dimensional space. Note that every two-dimensional node in Fig. 14 corresponds to several three-dimensional nodes; that is, to unclutter the figure, we enforce some condition in PRM construction such that some of the three-dimensional nodes differ only in the robot’s heading angle θ .

Belief b_∞ corresponding to each PRM node is also drawn. Indeed, we have $b_\infty = (\hat{x}_\infty^+, P_\infty)$. The 3σ covariance of P_∞ is shown by a black ellipse. The center of this ellipse has to be on \hat{x}_∞^+ , but because \hat{x}_∞^+ is random, we draw the P_∞ ellipse centered on the PRM node. For the \hat{x}_∞^+ , we show 3σ ellipse of its

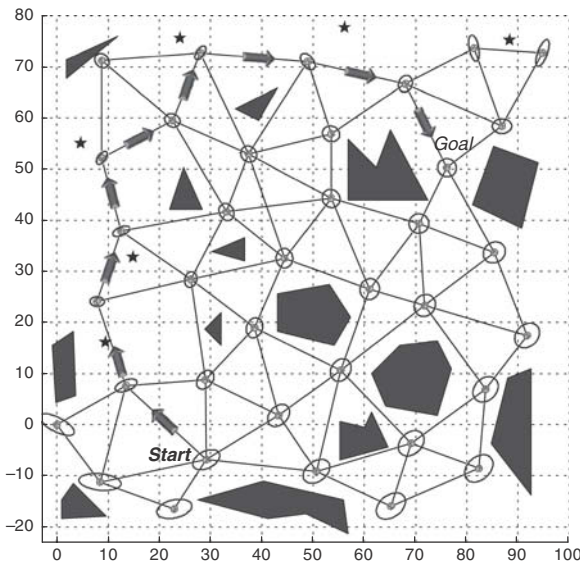


FIG. 14 This figure shows the two-dimensional projection of a three-dimensional PRM graph. As it is seen, the most likelihood path under the optimal policy detours from the shortest path towards the more informative region in the environment that leads to the safer control and obstacle avoidance probability.

covariance in red, that is, covariance of random vector $e^+ = \hat{x}_\infty^+ - \mathbf{v}$. Because $\mathbb{E}[\hat{x}_\infty^+] = \mathbf{v}$, this ellipse is also centered at \mathbf{v} . The green ellipse is the 3σ ellipse of covariance of x_∞ , that is, covariance of random vector $e = x_\infty - \mathbf{v}$.

Again, as it is seen from Fig. 14, the most likelihood path under the optimal policy detours from the shortest path towards the more informative regions in the environment. As a result, it reduces the collision probability and at the same time increases the estimation accuracy and controller efficiency.

G. CONCLUSION

In this part, we have proposed the feedback controller-based information-state roadmap (FIRM) for solving the motion-planning problem under motion and sensing uncertainties. This problem originally is a POMDP whose solution is intractable. Exploiting feedback controllers, we reduce it to a tractable FIRM MDP that can be solved by standard DP techniques. FIRM utilizes feedback controllers to create the reachable node regions in belief space and construct a graph on which a higher-level policy is defined to provide the optimal plans. An important consequence is that FIRM overcomes the curse of history in the original POMDP problem, as it preserves the optimal substructure property. Finally, by computing the collision probabilities, obstacles are also appropriately taken into account in planning on FIRM. By extending the probabilistic completeness concept to the planners under uncertainty, we also showed that FIRM is probabilistically complete under uncertainty. We believe that FIRM provides an important step toward solving POMDPs and utilizing them as a practical tool for robot motion planning under uncertainty.

APPENDIX A: PROOF OF LEMMA 4

Before proving Lemma 4, we state and prove following lemma

Lemma 5

Consider the bounded function $0 \leq f(\mathcal{X}) \leq 1$, and kernel $k(\mathcal{X}', \mathcal{X}) \geq 0$. Then, for any set \mathcal{A} , we have:

$$\left\| \int_{\mathcal{A}} k(\mathcal{X}', \mathcal{X}) f(\mathcal{X}') d\mathcal{X}' \right\| \leq \left\| \int_{\mathcal{A}} k(\mathcal{X}', \mathcal{X}) d\mathcal{X}' \right\| \quad (\text{A.1})$$

Proof: Given the properties of $f(\cdot)$ and $k(\cdot, \cdot)$, we have $k(\mathcal{X}', \mathcal{X}) f(\mathcal{X}') \leq k(\mathcal{X}', \mathcal{X})$, for all \mathcal{X} and \mathcal{X}' . Taking integral from both sides with respect to \mathcal{X}' and then taking supremum norm with respect to \mathcal{X} , the result follows. \square

Now, we prove Lemma 4.

Proof: If we denote the domain of operator \mathcal{T}_S by \mathcal{D} , we know that for all $f \in \mathcal{D}$, we have $0 \leq f(\mathcal{X}) \leq 1$, because $f(\mathcal{X})$ is the probability of some given set \mathcal{S} under some given controller invoked at point \mathcal{X} . Thus, it cannot be negative or greater than one, and based on Lemma 5, we have

$$\begin{aligned} \|\mathcal{T}_S[f]\| &= \left\| \int_{\bar{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) f(\mathcal{X}') d\mathcal{X}' \right\| \leq \left\| \int_{\bar{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' \right\| \\ &= \|\mathbb{P}_1(\bar{\mathcal{S}}|\mathcal{X}, \mu)\| \leq 1 \end{aligned} \quad (\text{A.2})$$

Therefore, based on the definition of operator norm, we have

$$\|\mathcal{T}_S\|_{\text{op}} = \sup_{f \neq 0} \{\|\mathcal{T}_S[f]\| : \forall f \in \mathcal{D}, \|f\| \leq 1\} \leq 1 \quad (\text{A.3})$$

According to Assumption 2, there exists a finite number N , such that

$$\inf_{\mathcal{X}} \mathbb{P}_n(\mathcal{S}|\mathcal{X}, \mu) = \beta > 0 \quad \forall n > N \quad (\text{A.4})$$

where “inf” and “sup” denote the infimum and supremum, respectively. Thus, we have

$$\begin{aligned} \|\mathbb{P}_n(\bar{\mathcal{S}}|\mathcal{X}, \mu)\| &= \sup_{\mathcal{X}} [1 - \mathbb{P}_n(\mathcal{S}|\mathcal{X}, \mu)] = 1 - \inf_{\mathcal{X}} \mathbb{P}_n(\mathcal{S}|\mathcal{X}, \mu) \\ &= 1 - \beta < 1 \quad \forall n > N \end{aligned} \quad (\text{A.5})$$

Let us denote the n th iterated kernel of \mathcal{T}_S as $p_n(\mathcal{X}'|\mathcal{X}, \mu)$. Because this iterated kernel is a pdf, we have $p_n(\mathcal{X}'|\mathcal{X}, \mu) \geq 0$, $\forall \mathcal{X}, \forall \mathcal{X}', \forall n$. We can write

$$\begin{aligned} \|\mathcal{T}_S^N[f]\| &= \left\| \int_{\bar{\mathcal{S}}} p_N(\mathcal{X}'|\mathcal{X}, \mu) f(\mathcal{X}') d\mathcal{X}' \right\| \\ &\leq \left\| \int_{\bar{\mathcal{S}}} p_N(\mathcal{X}'|\mathcal{X}, \mu) d\mathcal{X}' \right\| = \|\mathbb{P}_N(\bar{\mathcal{S}}|\mathcal{X}, \mu)\| \leq \alpha < 1 \end{aligned} \quad (\text{A.6})$$

where $\alpha = 1 - \beta$, and similar to Eq. (A.3), we get $\|\mathcal{T}_S^N\|_{\text{op}} \leq \alpha < 1$. From the operator norm properties, we have

$$\|\mathcal{T}_S^{N+1}\|_{\text{op}} \leq \|\mathcal{T}_S^N\|_{\text{op}} \|\mathcal{T}_S\|_{\text{op}} \leq \alpha < 1$$

and similarly for all $n \geq N$, we have

$$\|\mathcal{T}_S^n\|_{\text{op}} \leq \alpha < 1 \quad \forall n \geq N$$

Now, consider the series $\sum_{i=1}^{\infty} \|\mathcal{T}_S^i\|_{\text{op}}$. We can split the sum to smaller pieces as follows:

$$\sum_{n=1}^{\infty} \|\mathcal{T}_S^n\|_{\text{op}} = \sum_{n=1}^N \|\mathcal{T}_S^n\|_{\text{op}} + \sum_{i=1}^{\infty} \sum_{n=iN+1}^{(i+1)N} \|\mathcal{T}_S^n\|_{\text{op}}$$

But because $\|\mathbf{T}_{\mathcal{S}}^{n+1}\|_{\text{op}} \leq \|\mathbf{T}_{\mathcal{S}}^n\|_{\text{op}}$ for all $n \geq N$, we have

$$\sum_{n=iN+1}^{(j+1)N} \|\mathbf{T}_{\mathcal{S}}^n\|_{\text{op}} \leq N \|\mathbf{T}_{\mathcal{S}}^{iN}\|_{\text{op}}$$

Also, we know

$$\|\mathbf{T}_{\mathcal{S}}^{iN}\|_{\text{op}} \leq \|\mathbf{T}_{\mathcal{S}}^N\|_{\text{op}}^i \leq \alpha^i$$

and thus, we have

$$\begin{aligned} \sum_{n=1}^{\infty} \|\mathbf{T}_{\mathcal{S}}^n\|_{\text{op}} &= \underbrace{\sum_{n=1}^N \|\mathbf{T}_{\mathcal{S}}^n\|_{\text{op}}}_{\leq N} + \sum_{i=1}^{\infty} \sum_{n=iN+1}^{(i+1)N} \|\mathbf{T}_{\mathcal{S}}^n\|_{\text{op}} \\ &\leq N + \sum_{i=1}^{\infty} N \alpha^i = N + \frac{N}{1-\alpha} = \epsilon \leq \infty \end{aligned}$$

□

APPENDIX B. PROOF OF COROLLARY 1

Proof: We know $\|R\| \leq 1$, and thus we can write

$$\left\| \sum_{n=0}^{\infty} \mathbf{T}_{\mathcal{S}}^n[R] \right\| \leq \sum_{n=0}^{\infty} \|\mathbf{T}_{\mathcal{S}}^n\|_{\text{op}} \|R\| \leq \sum_{n=0}^{\infty} \|\mathbf{T}_{\mathcal{S}}^n\|_{\text{op}} \leq \epsilon < \infty$$

Thus, series $\sum_{n=0}^{\infty} \mathbf{T}_{\mathcal{S}}^n[R]$ is a convergent series, and we can define the operator $(I - \mathbf{T}_{\mathcal{S}})^{-1}[R] = \sum_{n=0}^{\infty} \mathbf{T}_{\mathcal{S}}^n[R]$. We have

$$\|(I - \mathbf{T}_{\mathcal{S}})^{-1}\|_{\text{op}} = \left\| \sum_{n=0}^{\infty} \mathbf{T}_{\mathcal{S}}^n \right\|_{\text{op}} \leq \epsilon < \infty \quad (\text{A.7})$$

□

APPENDIX C. PROOF OF PROPOSITION 1

Before proving Proposition 1, we state and prove a lemma on the continuity of the transition probability induced by the local controllers in its parameter.

Lemma 6

Given Assumption 3, there exists a $c_2 < \infty$ such that

$$\|p[\mathcal{X}'|\mathcal{X}, \mu(b; \mathbf{v})] - p[\mathcal{X}'|\mathcal{X}, \check{\mu}(b; \check{\mathbf{v}})]\| \leq c_2 \|\mathbf{v} - \check{\mathbf{v}}\| \quad (\text{A.8})$$

Proof: The result directly follows by combining two parts of Assumption 3. □

Now, we are ready to prove Proposition 1.

Proof: To show $\mathbb{P}(\mathcal{B}|\mathcal{X}, \mu)$ is continuous with respect to \mathbf{v} , we perturb \mathbf{v} to some $\check{\mathbf{v}}$ such that $\|\mathbf{v} - \check{\mathbf{v}}\| < r$. The local controller associated with node $\check{\mathbf{v}}$ is referred to as $\check{\mu}$, whose successful absorption region is denoted by $\check{\mathcal{B}}$ and stopping region is $\check{\mathcal{S}}$. Similarly, the corresponding transient operator and recurrent function are referred to as $\check{T}_{\check{\mathcal{S}}}$ and \check{R} . Finally, the success probability associated with the perturbed node $\check{\mathbf{v}}$ is $\mathbb{P}(\check{\mathcal{B}}|\mathcal{X}, \check{\mu})$. To shorten the statements, we refer to $\mathbb{P}(\mathcal{B}|\mathcal{X}, \mu)$ and $\mathbb{P}(\check{\mathcal{B}}|\mathcal{X}, \check{\mu})$ respectively by $\mathfrak{P}(\mathcal{X})$ and $\check{\mathfrak{P}}(\mathcal{X})$. As a result of node perturbation, the success probability is perturbed as

$$\begin{aligned} \mathbb{P}(\mathcal{B}|\mathcal{X}, \mu) - \mathbb{P}(\check{\mathcal{B}}|\mathcal{X}, \check{\mu}) &:= \mathfrak{P} - \check{\mathfrak{P}} = R + T_{\mathcal{S}}[\mathfrak{P}] - \check{R} - \check{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}] \\ &= R - \check{R} + T_{\mathcal{S}}[\mathfrak{P}] - T_{\mathcal{S}}[\check{\mathfrak{P}}] + T_{\mathcal{S}}[\check{\mathfrak{P}}] - T_{\check{\mathcal{S}}}[\check{\mathfrak{P}}] + T_{\check{\mathcal{S}}}[\check{\mathfrak{P}}] - \check{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}] \\ &= (R - \check{R}) + T_{\mathcal{S}}[\mathfrak{P} - \check{\mathfrak{P}}] + (T_{\mathcal{S}} - T_{\check{\mathcal{S}}})[\check{\mathfrak{P}}] + (T_{\check{\mathcal{S}}} - \check{T}_{\check{\mathcal{S}}})[\check{\mathfrak{P}}] \end{aligned}$$

where

$$T_{\check{\mathcal{S}}}[f(\cdot)](\mathcal{X}) := \int_{\check{\mathcal{S}}} p^{\mu}(\mathcal{X}'|\mathcal{X})f(\mathcal{X}') d\mathcal{X}' \quad (\text{A.9})$$

Let us define the operators $T_{\Delta\mathcal{S}} := (T_{\mathcal{S}} - T_{\check{\mathcal{S}}})$ and $\Delta T_{\check{\mathcal{S}}} := (T_{\check{\mathcal{S}}} - \check{T}_{\check{\mathcal{S}}})$. Now, based on Corollary 1, we can write

$$\mathfrak{P} - \check{\mathfrak{P}} = (I - T_{\mathcal{S}})^{-1} \left[R - \check{R} + T_{\Delta\mathcal{S}}[\check{\mathfrak{P}}] + \Delta T_{\check{\mathcal{S}}}[\check{\mathfrak{P}}] \right] \quad (\text{A.10})$$

and thus the following inequality holds on the supremum norm of the perturbation of the absorption probability:

$$\begin{aligned} \|\mathfrak{P} - \check{\mathfrak{P}}\| &\leq (I - T_{\mathcal{S}})^{-1} \|\mathfrak{P} - \check{\mathfrak{P}}\|_{\text{op}} (\|R - \check{R}\| + \|T_{\Delta\mathcal{S}}[\check{\mathfrak{P}}]\| + \|\Delta T_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\|) \\ &\leq c (\|R - \check{R}\| + \|T_{\Delta\mathcal{S}}[\check{\mathfrak{P}}]\| + \|\Delta T_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\|) \\ &= c (\|K_1(\mathcal{X})\| + \|K_2(\mathcal{X})\| + \|K_3(\mathcal{X})\|) \end{aligned} \quad (\text{A.11})$$

where $K_1(\mathcal{X}) := R(\mathcal{X}) - \check{R}(\mathcal{X})$, $K_2(\mathcal{X}) := T_{\Delta\mathcal{S}}[\check{\mathfrak{P}}(\cdot)](\mathcal{X})$, and $K_3(\mathcal{X}) := \Delta T_{\check{\mathcal{S}}}[\check{\mathfrak{P}}(\cdot)](\mathcal{X})$. In the following we bound K_1 , K_2 , and K_3 , and thus bound the $\|\mathfrak{P} - \check{\mathfrak{P}}\|$, accordingly.

I. BOUND FOR $K_1(\mathcal{X})$

The supremum norm of $K_1(\mathcal{X})$ is

$$\begin{aligned}
 \|K_1(\mathcal{X})\| &= \|R(\mathcal{X}) - \check{R}(\mathcal{X})\| \\
 &= \left\| \int_{\mathcal{B}} p^\mu(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' - \int_{\check{\mathcal{B}}} p^{\check{\mu}}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' \right\| \\
 &= \left\| \int_{\mathcal{B} \cap \check{\mathcal{B}}} [p^\mu(\mathcal{X}'|\mathcal{X}) - p^{\check{\mu}}(\mathcal{X}'|\mathcal{X})] d\mathcal{X}' \right. \\
 &\quad \left. + \int_{\mathcal{B} - \check{\mathcal{B}}} p^\mu(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' - \int_{\check{\mathcal{B}} - \mathcal{B}} p^{\check{\mu}}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' \right\| \\
 &\leq \int_{\mathcal{B} \cap \check{\mathcal{B}}} \|p^\mu(\mathcal{X}'|\mathcal{X}) - p^{\check{\mu}}(\mathcal{X}'|\mathcal{X})\| d\mathcal{X}' \\
 &\quad + \left\| \int_{\mathcal{B} - \check{\mathcal{B}}} p^\mu(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' + \int_{\check{\mathcal{B}} - \mathcal{B}} p^{\check{\mu}}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' \right\| \\
 &\stackrel{\text{from Eq. (A.8)}}{\leq} \int_{\mathcal{B} \cap \check{\mathcal{B}}} c_2 \|\mathbf{v} - \check{\mathbf{v}}\| d\mathcal{X}' + \|\mathbb{P}_1(\mathcal{B} \ominus \check{\mathcal{B}}|\mathcal{X}, \mu)\| \\
 &\quad + \|\mathbb{P}_1(\check{\mathcal{B}} \ominus \mathcal{B}|\mathcal{X}, \check{\mu})\| \\
 &\stackrel{\text{from Eq. (58)}}{\leq} c'_2 \|\mathbf{v} - \check{\mathbf{v}}\| + 2c' \|\mathbf{v} - \check{\mathbf{v}}\| = \gamma_1 \|\mathbf{v} - \check{\mathbf{v}}\| \tag{A.12}
 \end{aligned}$$

where $c'_2 < \infty$ and $\gamma_1 = c'_2 + 2c' < \infty$. In the penultimate inequality, we also used the fact that $\mathbb{P}_1(\check{\mathcal{B}} - \mathcal{B}|\mathcal{X}, \check{\mu}) \leq \mathbb{P}_1(\check{\mathcal{B}} \ominus \mathcal{B}|\mathcal{X}, \check{\mu})$ and $\mathbb{P}_1(\mathcal{B} - \check{\mathcal{B}}|\mathcal{X}, \mu) \leq \mathbb{P}_1(\mathcal{B} \ominus \check{\mathcal{B}}|\mathcal{X}, \mu)$ because $\check{\mathcal{B}} - \mathcal{B} \subseteq \check{\mathcal{B}} \ominus \mathcal{B}$ and $\mathcal{B} - \check{\mathcal{B}} \subseteq \mathcal{B} \ominus \check{\mathcal{B}}$.

II. BOUND FOR $K_2(\mathcal{X})$

We have

$$\begin{aligned}
 \|K_2(\mathcal{X})\| &= \|T_{\Delta\mathcal{S}}[\check{\mathfrak{P}}]\| = \|T_{\mathcal{S}}[\check{\mathfrak{P}}] - T_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\| \\
 &= \left\| \int_{\check{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' - \int_{\frac{\check{\mathcal{S}}}{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' \right\|
 \end{aligned}$$

$$\begin{aligned}
&= \left\| \int_{\bar{\mathcal{S}} - \check{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' - \int_{\check{\mathcal{S}} - \bar{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' \right\| \\
&\leq \left\| \int_{\bar{\mathcal{S}} - \check{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' + \int_{\check{\mathcal{S}} - \bar{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' \right\| \\
&= \left\| \int_{\bar{\mathcal{S}} \ominus \check{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' \right\| \stackrel{\text{from Eq. (A.1)}}{\leq} \left\| \int_{\bar{\mathcal{S}} \ominus \check{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' \right\| \\
&= \|\mathbb{P}_1(\bar{\mathcal{S}} \ominus \check{\mathcal{S}}|\mathcal{X}, \mu)\| \leq \|\mathbb{P}_1(\bar{\mathcal{B}} \ominus \check{\mathcal{B}}|\mathcal{X}, \mu)\| \\
&= \|\mathbb{P}_1(\mathcal{B} \ominus \check{\mathcal{B}}|\mathcal{X}, \mu)\| \stackrel{\text{from Eq. (5)}}{\leq} \gamma_2 \|\mathbf{v} - \check{\mathbf{v}}\| \tag{A.13}
\end{aligned}$$

where $\gamma_2 = \mathbf{c}' < \infty$. The penultimate inequality and equality follow from the relations $\bar{\mathcal{S}} \ominus \check{\mathcal{S}} \subseteq \mathcal{B} \ominus \check{\mathcal{B}}$ and $\bar{\mathcal{B}} \ominus \check{\mathcal{B}} = \mathcal{B} \ominus \check{\mathcal{B}}$, respectively.

III. BOUND FOR $K_3(\mathcal{X})$

We have

$$\begin{aligned}
\|K_3(\mathcal{X})\| &= \|\Delta \mathbf{T}_{\check{\mathcal{S}}}^{\check{\mathfrak{P}}}\| = \|\mathbf{T}_{\check{\mathcal{S}}}^{\check{\mathfrak{P}}} - \check{\mathbf{T}}_{\check{\mathcal{S}}}^{\check{\mathfrak{P}}}\| \\
&= \left\| \int_{\check{\mathcal{S}}} p^\mu(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' - \int_{\check{\mathcal{S}}} p^{\check{\mu}}(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' \right\| \\
&= \left\| \int_{\check{\mathcal{S}}} [p^\mu(\mathcal{X}'|\mathcal{X}) - p^{\check{\mu}}(\mathcal{X}'|\mathcal{X})] \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' \right\| \\
&\leq \int_{\check{\mathcal{S}}} \|p^\mu(\mathcal{X}'|\mathcal{X}) - p^{\check{\mu}}(\mathcal{X}'|\mathcal{X})\| \|\check{\mathfrak{P}}(\mathcal{X}')\| d\mathcal{X}' \\
&\stackrel{\text{from Eq. (A.8)}}{\leq} \int_{\check{\mathcal{S}}} c_2 \|\mathbf{v} - \check{\mathbf{v}}\| d\mathcal{X}' = \gamma_3 \|\mathbf{v} - \check{\mathbf{v}}\| \tag{A.14}
\end{aligned}$$

where $\gamma_3 < \infty$.

Therefore, based on Eqs.(A.11–A.14) we can conclude that

$$\|\mathbb{P}(\mathcal{B}|\mathcal{X}, \mu) - \mathbb{P}(\check{\mathcal{B}}|\mathcal{X}, \check{\mu})\| \leq \gamma \|\mathbf{v} - \check{\mathbf{v}}\| \quad (\text{A.15})$$

where $\gamma = c(\gamma_1 + \gamma_2 + \gamma_3) < \infty$, which completes the proof that absorption probability under the controller μ is continuous in the PRM node \mathbf{v} . \square

APPENDIX D. PROOF OF THEOREM 1

Before, starting with the proof of Theorem 1, we state the following proposition that concludes the continuity of the success probability of π (overall planner) given the continuity of the success probability of the individual local planners (μ^{ij} s).

Proposition A.1 (Continuity of success probability of π): The success probability $\mathbb{P}(\text{success}|b_0, \pi)$ is continuous in \mathcal{V} if the absorption probabilities $\mathbb{P}(B_j|b, \mu^j)$ is continuous in \mathbf{v}_j for all i, j , and b .

Proof: Given that $\mathbb{P}(B_j|b, \mu^j)$ is continuous with respect to \mathbf{v}_j for all i, j we want to show that $\mathbb{P}(\text{success}|\pi, b_0)$ is continuous with respect to all \mathbf{v}_j . First, let us look at the structure of the success probability.

$$\mathbb{P}(\text{success}|b_0, \pi) = \mathbb{P}(B(\mu_0)|b_0, \mu_0) \mathbb{P}(\text{success}|B(\mu_0), \pi^0) \quad (\text{A.16})$$

where μ_0 is computed using Eq. (34). The term $\mathbb{P}(B(\mu_0)|b_0, \mu_0)$ in the right-hand side of Eq. (A.16) is continuous because the continuity of $\mathbb{P}(B_j|b, \mu^j)$ for all i, j is assumed in this lemma. Thus, we only need to show the continuity of the second term in Eq. (A.16). Without loss of generality, we can consider $B_i = B(\mu_0)$. Then, it is desired to show that $\mathbb{P}(\text{success}|B_i, \pi^0)$ is continuous with respect to \mathbf{v}_i for all i .

As we saw in Sec. III.D, the probability of success from the i th FIRM node is as follows:

$$\mathbb{P}(\text{success}|B_i, \pi^0) = \Gamma_i^T (I - \mathcal{Q})^{-1} \mathcal{R}_g \quad (\text{A.17})$$

Moreover, we can consider $B_{\text{goal}} = B_N$ without loss of generality; then, the (i, j) th element of matrix \mathcal{Q} is $\mathcal{Q}[i, j] = \mathbb{P}(B_j|B_i, \pi^0(B_j))$, and the j th element of vector \mathcal{R}_g is $\mathcal{R}_g[j] = \mathbb{P}(B_N|B_j, \pi^0(B_j))$. Because we considered the B_j as the stopping region of the local controller μ^j , we have

$$\mathbb{P}(B_j|B_i, \mu^j) = 0 \quad \text{if } i \neq j \quad (\text{A.18})$$

Therefore, all of the nonzero elements in the matrices \mathcal{R}_g and \mathcal{Q} are of the form $\mathbb{P}(B_j|B_j, \mu^j)$. Thus, given the continuity of $\mathbb{P}(B_j|b, \mu^j)$, the transition probability $\mathbb{P}(B_j|B_j, \mu^j)$ are continuous, and the matrices \mathcal{R}_g and \mathcal{Q} are continuous. Therefore, $\mathbb{P}(\text{success}|B_i, \pi^0)$ and thus $\mathbb{P}(\text{success}|b_0, \pi)$ are continuous with respect to underlying PRM nodes. \square

Now, we are ready to prove Theorem 1.

Proof: Based on the definition of probabilistic completeness under uncertainty, if there exists a globally successful policy $\tilde{\pi}$, FIRM has to find a globally successful policy π as the number of FIRM nodes increases unboundedly. Thus, we start by assuming that there exists a globally successful

policy $\tilde{\pi} \in \Pi$. Because each policy in Π is parameterized by a PRM graph, there exists a PRM with nodes $\check{\mathcal{V}} = \{\check{\mathbf{v}}_i\}_{i=1}^N$ that parameterizes the policy $\tilde{\pi}$. Because $\tilde{\pi}$ is a globally successful policy, we know $\mathbb{P}(\text{success}|b_0, \tilde{\pi}) > p_{\min}$ for all $b_0 \in \mathbb{B}_{\Pi}$. Thus, we can define $\epsilon^* = \mathbb{P}(\text{success}|b_0, \tilde{\pi}) - p_{\min} > 0$.

Given Assumptions 2–4 and based on Propositions 1 and A.1, we know that $\mathbb{P}(\text{success}|b_0, \pi)$ is continuous with respect to the parameters of the local planners, that is, for any $\epsilon > 0$, there exists a $\delta > 0$, such that if $\|\mathcal{V} - \check{\mathcal{V}}\| < \delta$, then $|\mathbb{P}[\text{success}|b_0, \pi(\mathcal{V})] - \mathbb{P}[\text{success}|b_0, \tilde{\pi}(\check{\mathcal{V}})]| < \epsilon$. The notation $\|\mathcal{V} - \check{\mathcal{V}}\| < \delta$ means that $\|\mathbf{v}_i - \check{\mathbf{v}}_i\| < \delta$, for all i , or equivalently, $\mathbf{v}_i \in \check{\Omega}_i$, for all i , where $\check{\Omega}_i$ is a ball with radius δ , centered at $\check{\mathbf{v}}_i$.

Therefore, for the introduced ϵ^* , there exists a δ^* and corresponding regions $\{\check{\Omega}_i\}_{i=1}^N$, such that if we have a PRM whose nodes (or a subset of nodes) satisfy the condition $\mathbf{v}_i \in \check{\Omega}_i$ for all $i = 1, \dots, N$, then the planner π parameterized by this PRM has a success probability greater than p_{\min} , that is, $\mathbb{P}[\text{success}|b_0, \pi(\mathcal{V})] > p_{\min}$, and hence π is successful. (A subset of nodes is enough because the success probability is a nondecreasing function in terms of the number of nodes).

Because $\delta > 0$, the regions $\check{\Omega}_i$ have a nonempty interior. Consider a PRM with a sampling algorithm, under which there is positive probability of sampling in $\check{\Omega}_i$, such as uniform sampling. Thus, starting with any PRM, if we increase the number of nodes, a PRM node will eventually be chosen at every $\check{\Omega}_i$ with probability one. Therefore, the policy constructed based on these nodes will have a success probability greater than p_{\min} , that is, we eventually get a successful policy if one exists. Thus, FIRM is probabilistically complete. \square

REFERENCES

- [1] Bryson, A., and Ho, Y., *Applied Optimal Control*, AIAA, New York, 1979.
- [2] LaValle, S., *Planning Algorithms*, Cambridge Univ. Press, Cambridge, England, U. K., 2006.
- [3] Kavraki, L., Svestka, P., Latombe, J., and Overmars, M., “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces,” *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, 1996, pp. 566–580.
- [4] LaValle, S., “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” Computer Science Dept., Iowa State Univ., Tech. Rept. 98–11, Ames, IA, Oct. 1998.
- [5] Hsu, D., Kindel, R., Latombe, J., and Rock, S., “Randomized Kinodynamic Motion Planning with Moving Obstacles,” *The International Journal of Robotics Research*, Vol. 21, No. 3, 2002, p. 233.
- [6] Puterman, M., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, New York, 1994.
- [7] Bertsekas, D., *Dynamic Programming and Optimal Control*, Vols. 1 & 2, 2nd ed., Athena Scientific, Belmont, MA, 2000.
- [8] Bertsekas, D., and Tsitsiklis, J., *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996.
- [9] Sutton, R., and Barto, A., *Reinforcement Learning*, Vol. 18, MIT Press, Cambridge, MA, 1998.

- [10] Bertsekas, D., *Dynamic Programming and Optimal Control*, 3rd ed., Athena Scientific, Belmont, MA, 2007.
- [11] Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics*, MIT Press, Cambridge, MA, 2005.
- [12] LaValle, S., and Kuffner, J., "Randomized Kinodynamic Planning," *The International Journal of Robotics Research*, Vol. 20, No. 5, 2001, p. 378.
- [13] Sutton, R., Precup, D., and Singh, S., "Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning," *Artificial Intelligence*, Vol. 112, No. 1, 1999, pp. 181–211.
- [14] Parr, R., "Hierarchical Control and Learning for Markov Decision Processes," Ph.D. Dissertation, Univ. of California, Berkeley, CA, 1998.
- [15] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.
- [16] Sniedovich, M., "Dijkstra's Algorithm Revisited: the Dynamic Programming Connexion," *Control and Cybernetics*, Vol. 35, No. 3, 2006, pp. 599–620.
- [17] LaValle, S., "Robot Motion Planning: A Game-Theoretic Foundation," *Algorithmica*, Vol. 26, No. 3, 2000, pp. 430–465.
- [18] Amato, N., and Wu, Y., "A Randomized Roadmap Method for Path and Manipulation Planning," *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, 1996, Vol. 1, IEEE, Piscataway, NJ, 1996, pp. 113–120.
- [19] Guibas, L., Hsu, D., Kurniawati, H., and Rehman, E., "Bounded Uncertainty Roadmaps for Path Planning," *Algorithmic Foundation of Robotics VIII*, Vol. 57, Springer-Verlag, New York, 2008, pp. 199–215.
- [20] Burns, B., and Brock, O., "Sampling-Based Motion Planning with Sensing Uncertainty," *2007 IEEE International Conference on Robotics and Automation*, IEEE, Piscataway, NJ, 2007, pp. 3313–3318.
- [21] Missiuro, P., and Roy, N., "Adapting Probabilistic Roadmaps to Handle Uncertain Maps," *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 2006, ICRA 2006, IEEE, Piscataway, NJ, 2006, pp. 1261–1267.
- [22] Melchior, N., and Simmons, R., "Particle RRT for Path Planning with Uncertainty," *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, IEEE, Piscataway, NJ, 2007, pp. 1617–1624.
- [23] Wilmarth, S., Amato, N., and Stiller, P., "MAPRM: A Probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space," *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, Vol. 2, IEEE, Piscataway, NJ, 1999, pp. 1024–1031.
- [24] Prentice, S., and Roy, N., "The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance," *The International Journal of Robotics Research*, Vol. 28, Nos. 11–12, 2009, p. 1448.
- [25] Alterovitz, R., Branicky, M., and Goldberg, K., "Motion Planning Under Uncertainty for Image-Guided Medical Needle Steering," *The International Journal of Robotics Research*, Vol. 27, Nos. 11–12, 2008, p. 1361.
- [26] Alterovitz, R., Siméon, T., and Goldberg, K., "The Stochastic Motion Roadmap: A Sampling Framework for Planning with Markov Motion Uncertainty," *Robotics: Science and Systems*, edited by W. Burgard et al., MIT Press, Cambridge, MA, 2008, pp. 233–241.

- [27] Mellinger, D., and Kumar, V., "Control and Planning for Vehicles with Uncertainty in Dynamics," *2010 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ, pp. 960–965.
- [28] Pepy, R., Kieffer, M., and Walter, E., "Reliable Robust Path Planner," *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, IEEE, Piscataway, NJ, 2008, pp. 1655–1660.
- [29] Schouwenaars, T., Mettler, B., Feron, E., and How, J., "Robust Motion Planning Using a Maneuver Automation with Built-in Uncertainties," *American Control Conference, 2003. Proceedings of the 2003*, Vol. 3, IEEE, Piscataway, NJ, 2003, pp. 2211–2216.
- [30] Huang, Y., and Gupta, K., "RRT-SLAM for Motion Planning with Motion and Map Uncertainty for Robot Exploration," *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, IEEE, Piscataway, NJ, 2008, pp. 1077–1082.
- [31] Lazanas, A., and Latombe, J., "Motion Planning with Uncertainty: A Landmark Approach," *Artificial Intelligence*, Vol. 76, Nos. 1–2, 1995, pp. 287–317.
- [32] Censi, A., Calisi, D., Luca, A. D., and Oriolo, G., "A Bayesian Framework for Optimal Motion Planning with Uncertainty," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008.
- [33] Platt, R., Tedrake, R., Kaelbling, L., and Lozano-Perez, T., "Belief Space Planning Assuming Maximum Likelihood Observations," *Proceedings of Robotics: Science and Systems (RSS)*, 2010.
- [34] van den Berg, J., Abbeel, P., and Goldberg, K., "Lqg-mp: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information," *Proceedings of Robotics: Science and Systems (RSS)*, 2010.
- [35] van den Berg, J., Abbeel, P., and Goldberg, K., "Lqg-mp: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information," *International Journal of Robotics Research*, Vol. 30, No. 7, 2011, pp. 895–913.
- [36] Prentice, S., and Roy, N., "The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance," *International Journal of Robotics Research*, Vol. 8, Dec. 2009, pp. 1448–1465.
- [37] Huynh, V., and Roy, N., "iclgq: Combining Local and Global Optimization for Control in Information Space," *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.
- [38] Toit, N. D., and Burdick, J. W., "Robotic Motion Planning in Dynamic, Cluttered, Uncertain Environments," *2010 IEEE International Conference on Robotics and Automation (ICRA)*, 2010.
- [39] He, R., Brunskill, E., and Roy, N., "Efficient Planning Under Uncertainty with Macro-Actions," *Journal of Artificial Intelligence Research*, Vol. 40, Feb. 2011, pp. 523–570.
- [40] Kurniawati, H., Du, Y., Hsu, D., and Lee, W. S., "Motion Planning Under Uncertainty for Robotic Tasks with Long Time Horizons," *International Journal of Robotics Research*, Vol. 30, No. 3, 2011, pp. 308–323.
- [41] Kavraki, L., Kolountzakis, M., and Latombe, J., "Analysis of Probabilistic Roadmaps for Path Planning," *IEEE Transactions on Robotics and Automation*, Vol. 14, Feb. 1998, pp. 166–171.
- [42] Ladd, A., and Kavraki, L., "Measure Theoretic Analysis of Probabilistic Path Planning," *IEEE Transactions on Robotics and Automation*, Vol. 20, April 2004, pp. 229–242.

- [43] Švestka, P., and Overmars, M., "Motion Planning for Car-Like Robots Using a Probabilistic Learning Approach," *International Journal of Robotics Research*, Vol. 16, No. 2, 1997, pp. 119–143.
- [44] Bohlin, R., "Robot Path Planning," Ph.D. Dissertation, Dept. of Mathematics, Chalmers Univ. of Technology, Goteborg, Sweden, 2002.
- [45] Hsu, D., "Randomized Single-Query Motion Planning in Expansive Spaces," Ph.D. Dissertation, Dept. of Computer Science, Stanford Univ., Stanford, CA, 2000.
- [46] Kavraki, L., Latombe, J., Motwani, R., and Raghavan, P., "Randomized Query Processing in Robot Motion Planning," *Proceedings of the ACM Symposium Theory of Computing*, ACM Press, Las Vegas, 1995, pp. 353–362.
- [47] Astrom, K., "Optimal Control of Markov Decision Processes with Incomplete State Estimation," *Journal of Mathematical Analysis and Applications*, Vol. 10, No. 1, 1965, pp. 174–205.
- [48] Smallwood, R. D., and Sondik, E. J., "The Optimal Control of Partially Observable Markov Processes over a Finite Horizon," *Operations Research*, Vol. 21, No. 5, 1973, pp. 1071–1088.
- [49] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R., "Planning and Acting in Partially Observable Stochastic Domains," *Artificial Intelligence*, Vol. 101, No. 1-2, May 1998, pp. 99–134.
- [50] Pineau, J., Gordon, G., and Thrun, S., "Point-Based Value Iteration: An Anytime Algorithm for POMDPs," *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2003, pp. 1025–1032.
- [51] He, R., Brunskill, E., and Roy, N., "PUMA: Planning Under Uncertainty with Macro-Actions," *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI)*, 2010.
- [52] Isaacson, D., and Madsen, R., *Markov Chains, Theory and Applications*, Vol. 4, Wiley, New York, 1976.
- [53] Norris, J. R., *Markov Chains*, Cambridge Univ. Press, Cambridge, England, U. K., 1999.
- [54] Brockett, R., "Asymptotic Stability and Feedback Stabilization," *Differential Geometric Control Theory*, edited by R. W. Brockett, R. S. Millman, and H. J. Sussmann, Birkhauser, Boston, 1983, pp. 181–191.
- [55] Au, S., and Beck, J., "Subset Simulation and Its Application to Seismic Risk Based on Dynamic Analysis," *Journal of Engineering Mechanics*, Vol. 129, No. 8, 2003, p. 901.
- [56] Kolmanovsky, I., and McClamroch, N., "Developments in Nonholonomic Control Problems," *Control Systems Magazine*, Vol. 15, No. 6, 1995, pp. 20–36.
- [57] Oriolo, G., De Luca, A., and Vendittelli, M., "WMR Control Via Dynamic Feedback Linearization: Design, Implementation, and Experimental Validation," *IEEE Transactions on Control Systems Technology*, Vol. 10, No. 6, 2002, pp. 835–852.
- [58] Campion, G., Bastin, G., and Dandrea-Novet, B., "Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 1, 1996, pp. 47–62.
- [59] Kumar, S., and Chakravorty, S., "Adaptive Sampling for Generalized Sampling Based Motion Planners," *2010 49th IEEE Conference on Decision and Control (CDC)*, IEEE, Piscataway, NJ, pp. 7688–7693.
- [60] Papadimitriou, C., and Tsitsiklis, J. N., "The Complexity of Markov Decision Processes," *Mathematics of Operations Research*, Vol. 12, No. 3, 1987, pp. 441–450.

- [61] Madani, O., Hanks, S., and Condon, A., "On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems," *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI)*, 1999, pp. 541–548.
- [62] Kavraki, L., Svestka, P., Latombe, J., and Overmars, M., "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, 1996, pp. 566–580.
- [63] Amato, N., Bayazit, B., Dale, L., Jones, C., and Vallejo, D., "Obprm: An Obstacle-Based PRM for 3D Workspaces," *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, 1998, pp. 155–168.
- [64] Lavalle, S., and Kuffner, J., "Randomized Kinodynamic Planning," *International Journal of Robotics Research*, Vol. 20, No. 5, 2001, pp. 378–400.
- [65] Pineau, J., Gordon, G., and Thrun, S., "Anytime Point Based Approximations for Large Pomdps," *Journal of Artificial Intelligence Research*, Vol. 27, 2006, pp. 335–380.
- [66] Spaan, M., and Vlassis, N., "Perseus: Randomized Point-Based Value Iteration for Pomdps," *Journal of Artificial Intelligence Research*, Vol. 24, 2005, pp. 195–220.
- [67] Bertsekas, D., *Dynamic Programming and Stochastic Control*, Academic International Press, New York, 1976.
- [68] Agha-Mohammadi, A., Chakravorty, S., and Amato, N., "FIRM: Feedback Controller-Based Information-State Roadmap, a Framework for Motion Planning Under Uncertainty," Parasol Lab., Dept. of Computer Science, Texas A&M Univ., Technical Rept. TR11-001, College Station, TX, Jan. 2011.
- [69] Agha-Mohammadi, A., Chakravorty, S., and Amato, N., "FIRM: Feedback Controller-Based Information-State Roadmap -a Framework for Motion Planning Under Uncertainty-," International Conference on Intelligent Robots and Systems (IROS), 2011.
- [70] Agha-Mohammadi, A., Chakravorty, S., and Amato, N., "On the Probabilistic Completeness of the Sampling-Based Feedback Motion Planners in Belief Space," IEEE International Conference on Robotics and Automation (ICRA), 2012.
- [71] Keener, J. P., *Principles of Applied Mathematics: Transformation and Approximation*, 2nd ed., Westview Press, 2000.
- [72] Resnick, S. I., *A Probability Path*, Birkhäuser, Boston, Cambridge, MA, 1999.
- [73] Kalmár-Nagy, T., D'Andrea, R., and Ganguly, P., "Near-Optimal Dynamics Trajectory Generation and Control of an Omnidirectional Vehicle," *Robotics and Autonomous Systems*, Vol. 46, Jan. 2004, pp. 47–64.
- [74] Crassidis, J., and Junkins, J., *Optimal Estimation of Dynamic Systems*, Chapman & Hall/CRC Press, Boca Raton, FL, 2004.
- [75] Simon, D., *Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches*, Wiley, New York, 2006.
- [76] Doucet, A., de Freitas, J., and Gordon, N., *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, New York, 2001.

Protocol Utilization in Intelligent Systems to Facilitate Exploration Missions

M. Lyell^{*} and W. Drozd[†]

Intelligent Automation, Inc., Rockville, Maryland 20855

A. Grinberg Webb[‡]

Montgomery College, Rockville, Maryland 20850

J. Nanda[†] and W. Chen[†]

Intelligent Automation, Inc., Rockville, Maryland 20855

I. INTRODUCTION

The focus of this work concerns protocol development in order to support interactions among autonomous entities engaged in exploration missions. The autonomous entity types include humans (astronauts or mission control), an individual software agent (softbot), multi-agent systems (MAS), robots, and autonomous robot teams. We investigate how properly designed protocols for use among the mixed autonomous entity types can provide structured but flexible interactions that reflect mission norms. The resulting protocol set provides the basis for developing a framework for human-autonomous system interactions.

Development for future space exploration missions is increasingly focused on autonomous system and robotic support. Multi-agent systems have been proposed for use in controlling environmental and life support systems on long-duration missions [1]. Studies have also considered agent-based systems for the task of health monitoring of physical hardware, such as onboard power systems [2]. Robots have been utilized in scientific missions on Mars. Mobile robots and robot teams have been proposed for lunar habitat development and for scientific sampling missions [3, 4]. In the general case, multi-agent systems (also termed softbots), mobile robots, and humans will interact to support the mission needs. Technologies that have contributed to specific applications in these areas include the BRAHMS [5] system for task analysis and simulation as well as

^{*}Principal Scientist; currently Visiting Researcher, The Samraksh Company, Leesburg, VA 20176; mltech2011@yahoo.com.

[†]Lead Scientist.

[‡]Professor.

planners and task schedulers, for example, ASPEN and CASPER [6], to support mission activities.

The modes of interaction among the autonomous entities will vary. In some cases, a multi-agent system may utilize services offered by another multi-agent system. An example is a vehicle-wide health monitoring agent team that takes as input the assessments developed by a second multi-agent team regarding the status of a vehicle's power system. Alternatively, a dialogue-capable softbot with knowledge of replacement procedures may assist an astronaut as he or she changes a unit on-orbit.

Although the preceding examples all involved software-based agents, robots also have a significant role in exploration missions. Robotic assistants such as the Personal Satellite Assistant (PSA) (data available online at <http://psa.arc.nasa.gov/about.shtml>) can utilize their mobility and their cameras to provide the astronaut with more information during an extra-vehicular activity on orbit. On the surface of the moon, mobile robots could directly assist astronauts with communications and habitat infrastructure development and maintenance as well as science missions. Modes of astronaut-to-robot interactions have been investigated. Fong and colleagues [7] have developed a human-robot interaction architecture that abstracted the robots as agents; however, their work included the notion of a server for the agents, rather than direct interactions among robot peers.

A second approach to astronaut-robot interaction, considered in the work of Hirsh and colleagues [8], involves the astronaut directly managing one or more robots. One alternative that is explored in this work involves the astronaut in a role of providing oversight to the actions of an autonomous team of robots engaged in task activity. In this work, the tasks are pertinent in the context of a notional scientific sampling mission in a lunar environment [3]. Although the robots act autonomously, an astronaut may traverse the mission area and interact with a robot directly regarding its status.

Previous research has also addressed the use of both robots and software agent-based autonomous entities to form teams with astronauts in space mission activities. Notable among these are the work of Bradshaw et al. [9] in the area of teamwork-centered autonomy and Schreckenghost et al. [10] in the area of advanced life support systems.

Whether on long-duration missions or on lunar surface excursions, interactions between autonomous entities and astronauts must include an emphasis on safety. Protocol development must support safety. The concept of a lunar scientific sampling mission has been of great interest. Allen et al. [3] and Shearer and Neal [4] discuss scientific sampling missions that will occur as part of lunar exploration. Fong and colleagues [11] discuss site survey and the use of robots in sampling missions. Allen and colleagues provide information on proper curation of lunar samples. Among their recommendations is that "Upon collection a geological sample should be split into subsamples for preliminary examination at the lunar outpost, detailed analyses on Earth, and minimally contaminated storage on the Moon" [3]. Shearer and Neal note that "Sample return and

sample science have a substantial role to play in the human exploration and eventual habitation of the Moon" [4]. Furthermore, they explicitly state that robots have a role in sample return "either prior to or in conjunction with initial human sorties" [4]. Fong and Nourbakhsh [12] note that "robots will be used for a wide range of tasks The intricate nature of these tasks may require teams of robots capable of monitoring their own progress with a high level of autonomy." They also note that human-robot interaction will occur in various modes; ground control can direct robot teams, and astronauts can "communicate with in-situ robots" [12], and robot teams can interact in support of a common goal.

A scenario-driven approach was adopted in investigating autonomous entity interaction design and protocol support. Two primary scenarios were investigated: 1) a long-duration mission involving an astronaut crew member onboard (either orbiting or in-transit) and the actions of multiple soft-bots (multi-agent systems), each addressing a specific functionality pertinent to the vehicle subsystems; and 2) a team of autonomous robots that are engaged in a lunar scientific sampling mission in which the Astronaut can provide oversight.

The software agent paradigm is utilized as the abstraction for the autonomous entities in both scenarios. Although this is expected in interactions involving multi-agent systems, in the case of the mobile robots, each robot is considered as a robot agent, that is, an individual agent with physical characteristics. We note that the introduction of a protocol-awareness capability into a robotics framework that utilized a robot agent required an enhancement of that robot agent's cognitive capabilities.

Investigation of the long-duration mission scenario focused on how the astronaut could provide oversight of the multi-agent teams. This led to consideration of what information should be exposed by each multi-agent team. Multi-agent teams could be interdependent. Previous research that focused on situations in which humans interact with autonomous entities introduced the notion of an *autonomy level*; Dorais and Kortenkamp [13] have introduced the concept of "adjustable autonomy." In the long-duration mission scenario, the possibility of an astronaut-ordered agent team reconfiguration to a partial autonomy state could affect the behavior of other multi-agent teams. The astronaut would need to know the ramifications of changes to the autonomous activity that is being carried out by a software agent system and/or robotic system.

Dorais and Kortenkamp note that with adjustable autonomy, autonomous systems will need to operate at different levels of independence, intelligence, and control [13]. To support this, they note that 1) any autonomous system's autonomy level should be able to be changed by an "external agent" and 2) there should be clear, unambiguous protocols utilized for changing an autonomy level. The view of this work is that protocols are a key feature of any framework that supports human-autonomous system interaction.

Considerations of mission behavior/norms in both scenario-based investigations affect protocol development. Norms involve safety concerns and the need for mission success. Safety concerns clearly arise in the case of physical robots

vis-à-vis nearby astronauts in the lunar scientific sampling mission. However, safety is also an issue with interdependent multi-agent systems. Unintended consequences may result if performance is degraded in one agent-based subsystem due to a lowered autonomy mode in another agent-based subsystem.

The chapter is organized as follows. Section II provides a discussion of mission criteria and behavioral expectations. Section III provides background in agent technology and describes the nature of “protocol” in the context of this work. Section IV addresses the protocol development and framework features that were motivated by the long-duration mission scenario. Section V is focused on the robot agent and its cognitive enhancement as preparation for protocol development. Section VI addresses the protocol development and framework features that were motivated by the Lunar Scientific Sampling Mission. Section VII presents results on a deployed autonomous robot team in the laboratory environment, engaged in the scientific sampling mission. Section VIII discusses the results and presents conclusions. Portions of this work are found in [14–16].

II. MISSION CRITERIA AND BEHAVIOR NORMS: THE IMPACT ON FRAMEWORK DESIGN

Protocol development, which is motivated by the need to support interactions between astronauts and the multi-agent system “crew” or between astronauts and robots or among the autonomous entities, is grounded in expectations about such interactions. To provide this grounding, the concept of a *Mission Society* is introduced. Within the context of a Mission Society, *stakeholders, actors, responsibilities and expectations of behavior* are identified. Mission needs are driven by notional lunar scientific sampling mission and long-duration mars mission concepts. Mission Society behavior norms will guide protocol development. Of particular interest are Mission Society behavior characteristics that pertain to information exposure, astronaut safety, and expectation of response.

A. MISSION SOCIETY AND BEHAVIOR NORMS

1. MISSION SOCIETY NORMS

The Mission Society is defined in general as a mission-oriented society involving human, multi-agent systems, robot, and robot team participants. This abstraction provides the overarching context in which a multi-agent system that is participating in a long-duration exploration mission or in which an autonomous robot team that is executing a scientific sampling mission must operate. *Stakeholders* in a Mission Society extend beyond those of the individual agents or robots or astronauts. Actors include 1) mission control, 2) astronauts/human crew, 3) multi-agent systems that operate to support the

mission, 4) individual agents within a MAS, 5) robot agent team, and 6) individual robot agents.

Primary *values* of the Mission Society were identified as critical value placed on human crew safety and high value placed on mission completion and success.

The question of behavior norms in long-duration missions is the subject of current research [17]. We restrict our attention to behavior norms that are reflected in standard operating procedures, with a focus on norms related to conveying information, safety, and clear organization/roles.

2. BEHAVIOR NORMS

- With this restriction, primary *behavioral expectations or norms* relevant within the Mission Society include the following: 1) behavior norm of option assessment or activities vis-à-vis classification schemes, for example, critical function, priority levels; 2) behavior norm of methodical processes, for example, use of procedures to structure activity and interactions with (nonautonomous) machinery; role-based behavior within team, subject to a command structure; and behavior norm of safety within methodical processes.

Such norms lead to the following ramifications regarding how autonomous entities such as multi-agent systems and robot teams could participate in the Mission Society. These include the following:

- There is a need for a multi-agent system to provide access to information on its subsystem functioning and any changes.
- There is a need for an autonomous entity team to utilize relevant information to guide its activity selection.
- There is a need for a robot team to provide information on its task status.
- There is a need for a representative of the autonomous entity to interact with the astronaut crew, leading to role-based capability within the autonomous entity teams.
- A multi-agent system or robot team has the ability to relinquish its autonomous activity at the command of an astronaut.
- There is a need for safety in autonomous team functioning, especially when changing autonomy levels or, in the case of a robot team, when physically near to a human.
- Teamwork orientation, with roles and potential command structure within an autonomous system, is necessary.

To summarize, the behavior that must be engaged in by the autonomous teams is one of 1) methodical procedures and 2) information utilization with critical

assessment in order to ensure crew safety and enhance the probability of mission success. The behavior pattern exists within a teamwork environment, modified by a command structure. Framework design to support astronaut and multi-agent autonomous system interactions should be informed by behavioral expectations.

B. IMPACT ON FRAMEWORK DESIGN

Activities within the Mission Society can vary as to the nature of the lead entity, the participants, and the scope of the activity that is organized. Table 1 presents various options.

Mixed teams comprised of humans and multi-agent systems can interact within the context of different modes of task organization. For example, a task may be decomposed into multiple subtasks, each of which is handled by a single entity. A software agent or multi-agent system (MAS) team may handle a portion of the subtasks, while humans handle the remaining subtasks. In this case, there may not be direct human–software agent interaction. Alternatively, an astronaut may be in charge of all subtask activities, including those performed by MAS as well as other humans. The task decomposition process can produce tasks that must be handled by multiple entities. In this case, a human may interact with members of a MAS team to achieve the task goal. Note that although a MAS is composed of multiple software agents, an astronaut may interact only with a particular subset of these agents.

An astronaut who is leading an activity will likely be considered as unavailable for other activities. The astronaut will not welcome interruptions by others, whether human or MAS. Human members of the crew will recognize this and adjust their interactions accordingly. A MAS activity should also not interrupt an astronaut that is currently engaged, unless necessary. This can adversely impact safety.

For example, MAS team members should not have unfettered initiative in establishing a conversation with the astronaut. Hexmoor and Vaughn [18] divide adjustable autonomy into the dynamic and deterministic; the deterministic autonomy pertains to “the agent’s ability to refrain from actions it can perform.”

The need for adjustable autonomy also may be motivated by nontechnical reasons [13], with “a desire to allow for human intervention even when full autonomy is possible. These motivations include safety, training, maintenance or calibration.” As part of the Mission Society, the astronaut must be able to command a MAS to change its autonomy level.

Astronaut interaction with autonomous robot team entities involves additional safety issues that arise because of mobility, robot size, and the nature of the lunar environment. The physical proximity of a robot member of the team is a factor that must be taken into account when developing protocols that support interaction.

TABLE 1 POTENTIAL PARTICIPANTS IN MISSION SOCIETY INTERACTIONS

Lead Actor	Role	Additional Participants	Examples of Utilization
Astronaut	In charge, solo		Specific, focused activity
Astronaut	In charge of activity	Multi-agent system	Focused activity with software agent participation
Robot	Team leader (in charge of mission area)	Robots on team	Lunar scientific sampling mission
Astronaut	Oversight	Team of robots	Oversight of autonomous robot team engaged in lunar scientific sampling mission
Astronaut	In-charge of human team composed of astronauts	Astronauts participating as team members	Space-based and space-commanded team activity
Astronaut	In-charge of human and MAS team members	Mixed initiative team	More complex space-based and space-commanded team activity, virtual teams
Astronaut	In-charge of multiple MAS team members	Each MAS has participating software agents	Astronaut has oversight of the actions of multiple autonomy applications
Mission control and astronaut team	In-charge of human team composed of astronauts	Astronauts participating as team members	Ground-commanded / controlled task set, with space-based task action involving team activity
Mission control and astronaut team	In-charge of mixed team composed of astronauts and (multiple) MAS	Astronauts and MAS, participating as team members. Astronaut may be in charge of a specific activity with a MAS team member.	Ground-commanded / controlled task set, with space-based task action involving team activity. The team is mixed.

III. TECHNOLOGY FUNDAMENTALS: SOFTWARE AGENT PARADIGM AND PROTOCOLS

A. PROTOCOL

A protocol can be thought of as a structured conversation. One representation of a protocol is via a finite state machine. The conversation exchanges are thus bounded, but each exchange is *not* deterministic. Each conversational exchange, or message, involves a “performative.” A conversation can be viewed as an instance of a protocol that involves the use of a set of performatives. The concept of the performative is grounded in speech act theory developed by, among others, Searle [19].

The performative indicates the semantic sense for the type of message content that should be sent at that point in the protocol. For example, if a “query” performative is used, then the message content should involve some question. Semantically meaningful and nonambiguous performatives have been standardized by the Foundation for Intelligent Physical Agent’s (FIPA) [20]. These include “agree” and “inform-done.”

Kremer and Flores [21], in research on social commitment and protocol usage, have noted that message responses that serve to acknowledge a communication indicate the presence of social commitment among the participating autonomous entities of a system. Furthermore, Kremer and Flores have reviewed the standardized FIPA protocols; their analysis has provided a mapping that indicates that the standardized FIPA protocols meet the needs of social commitment.

With regard to the detailed structure of the protocol, each protocol can be viewed as a finite state machine. An allowable interaction within the protocol is denoted in terms of its {performative} or a {performative, context} pair. The protocol can be viewed as a finite state machine. The selection by an agent (or robot agent) of a permitted response is chosen from among a set of allowable response types, subject to guard conditions that depend on the situation. An instance of a conversation that is based on a protocol is bounded but not deterministic. The guard conditions depend on parameters relevant to the domain and can include information on the situation of other agents, whether robot agents or soft-bot agents.

B. SOFTWARE AGENT PARADIGM

Software agents are software components that have specific characteristics which provide them with a sense of agency. The software agent paradigm identifies the agent characteristics as 1) having a goal orientation and being proactive, 2) being communicative with other agents, and 3) having adaptability or flexibility. Typically, an application involves the actions of multiple agents, organized into a multi-agent system. The communicativeness property supports message exchange among the agents that must interact. Protocol development serves to organize such interactions in a flexible manner. A deeper discussion of the software agent paradigm is provided by Wooldridge and Jennings [22].

Jennings [23] notes that the software agent paradigm is ideal for use in developing complex, loosely coupled systems. Software development to support multiple interacting subsystems on long-duration missions that can interact asynchronously with each other as well as with an astronaut is one example of a complex, loosely coupled system. An agent-oriented software engineering (AOSE) process such as Gaia [24] can be utilized in order to design each multi-agent system. Griss and Pour [25] have referred to software agents as representing the future of components.

The use of the software agent paradigm in robotics allows a robot to be treated as an individual entity capable of agency. Robot control software that reflects the agent paradigm is useful in developing entities that participate in an autonomous team. In particular, the communicativeness property of agents directly supports the development of coordination mechanisms among autonomous robots engaged in teamwork activity.

IV. MULTI-AGENT-BASED SUBSYSTEMS AND ASTRONAUT INTERACTIONS AND OVERSIGHT: BEHAVIOR AND PROTOCOL DEVELOPMENT

The Mission Society abstraction and its behavior norms motivate elements of the software framework design to support astronaut and multi-agent autonomous system interactions. Behavior norms specifically affect information exposure and utilization decisions as well as autonomy level change and safety in interactions between agents and humans.

With regard to information exposure within the framework: In current space missions, information regarding subsystem functioning and subsystem status changes is typically reviewed by humans. Information on subsystem functioning is provided by specific sensors whose data are sent via telemetry to Mission Control. In the event of abnormal behavior onboard current crewed space missions, this information is accessed and reviewed by trained Mission Control operators who are knowledgeable on subsystem dependencies and error modes. Further investigation is done by a team collaboration of astronaut crew and Mission Control members; this may include subsystem mode changes that are performed manually by an astronaut.

The addition of autonomous systems for sensor information assessment is a planned element of long-duration missions; however, astronaut crew and Mission Control personnel would still participate in assessment. Although a multi-agent system or software agent may be in charge of a specific activity, the human crew and/or Mission Control personnel will retain the ability to modify the MAS activity. This oversight and control may be managed through an implementation of the concept of adjustable autonomy [13], which is done in accordance with the norms of the Mission Society.

The existence of autonomous entities involved in subsystem management can lead to complications. Changed sensor readings may not be a result of a subsystem

error, but may result from changes that the controlling MAS made in response to perceived conditions.

To comply with Mission Society norms pertaining to 1) information access and utilization and 2) safety, each autonomous multi-agent system must do the following: expose its dependencies on other MAS or subsystems, expose its operational status, expose safety-related information, and expose information on current autonomy mode. Thus, to support behavior patterns of the Mission Society regarding human–software agent collaborations, each multi-agent autonomous system must do the following:

- It must respond to astronaut's commands to change its autonomy level.
- Generally, MAS is to refrain from taking dialogue initiative with astronaut.
- Interactions should be supported by methodical protocols.
- Protocol design is to reflect “commitment” in support of Mission Society expectations.

A. MOTIVATING SCENARIOS

A set of use case scenarios involving five exemplar multi-agent systems was considered. Functionality for each MAS was defined. Not all MAS had functionality that spanned all levels of adjustable autonomy. The *overarching use case* is composed of a union of the five use cases. It may be described as one involving the activity of multiple MAS performing their functions in an autonomous manner, when the need for specific astronaut activity, that of changing an LRU that is connected to the power system, impacts these MAS. Before proceeding with the replacement, the astronaut must review MAS functionality per their different autonomy modes, as well as MAS dependencies. Consequently, the astronaut must command multiple MAS to change their autonomy mode.

Each individual use case scenario focuses on the functionality of a particular MAS. The details of each of the use cases were designed to require that each MAS utilize the services of another. This increases the potential for needed interactions.

Particular key elements of the overarching scenario include 1) virtual team formation, 2) direct astronaut-MAS dialogue, and 3) degradation of health monitoring results due to lack of information from another MAS whose autonomy mode was ordered to “none” by the astronaut. Reference [26] provides much detail on each of the individual MAS use cases. These use case scenarios were divided into two groups, A and B; see Table 2.

All MAS use cases were analyzed in order to determine the information exposure needed to support interactions consistent with Mission Society expectations.

The Group B MAS Use Cases were prototyped in software. The agent implementation for each MAS was developed in Java and executed on the CybeleTM agent development and execution platform [23].

TABLE 2 INDIVIDUAL USE CASE SCENARIOS

Use Case Designator	Group	Description
<i>Group A</i>		
Use Case A_1		Multi-agent autonomous system supports the sending of optional telemetry data. <i>This is abbreviated as the Comms Tel MAS.</i>
Use Case A_2		Multi-agent autonomous system provides medical monitoring of astronaut by software autonomy application team.
<i>Group B</i>		
Use Case B_1		Multi-agent autonomous system provides monitoring of the physical health of hardware systems [(<i>Hardware Health MAS</i>).
Use Case B_2		Multi-agent autonomous system controls access to final stage power distribution (to switches) and provides information on power switch settings.
Use Case B_3		Astronaut and multi-agent autonomous system's dialogue team member (an autonomy component) replace a line replaceable unit (LRU).

B. BEHAVIOR REQUIREMENTS AND ROLE CONSTRUCTS

Tables 3 and 4 elucidate the detailed requirements that are placed on each multi-agent system that wishes to participate in the Mission Society behavior norm-compliant framework. These are categorized into four groups: 1) information exposure (IE), 2) functionality for multiple modes of operation (F), 3) interaction (INT), and 4) visualization and reporting (VR).

With reference to Table 3, note that the *coordinator (or lead) role* is a nexus of information flow activities. This role is then an excellent candidate to also assume the responsibility of handling the autonomy change protocol in support of the Mission Society needs.

An outcome of the iterated AOSE analysis phase is that a new role related to supporting the interactions between MAS and an astronaut has been identified. The *designated/designee* role is introduced; this role must engage in direct communication with the astronaut when the autonomy level is nonautonomous. That is, the particular software entity (agent) that embodies the designated role must offer a specific capability that the astronaut can invoke directly.

TABLE 3 INFORMATION EXPOSURE REQUIREMENTS ON A MAS IN ORDER TO PARTICIPATE IN THE MISSION SOCIETY

Area of Applicability	Requirement
Information exposure	REQ IE1: Each MAS must offer a protocol for changing from one autonomy mode to another. The protocol must support: <ul style="list-style-type: none">○ Informing current clients of the system of the autonomy mode change and what services or products are no longer available○ Informing past MAS clients of the system of the autonomy mode change what services or products are no longer available○ Informing client coordinators and the astronaut of any safety ramifications○ Acknowledgments must be incorporated into the protocol design
Information exposure	REQ IE 2: Each MAS must “self-describe” its <ul style="list-style-type: none">○ Products or services under an autonomy level change○ Products or services under an operational change (which may be caused by a change in the autonomy level of a MAS from which it needs a product or service to produce its product or service)
Information exposure	REQ IE 3: Each MAS must provide, upon startup and upon changes, <ul style="list-style-type: none">○ A list of its subautonomy components and their active status○ Its dependency network (on other MAS)○ The level of its current autonomy mode
Information exposure	REQ IE 4: Each MAS that receives a message informing it of another MAS’s autonomy change must immediately <ul style="list-style-type: none">○ Update its dependency network○ Make a change in its operational status if needed information or services becomes unavailable○ Propagate the operational status change○ Propagate any safety alerts

For example, in the *Comms Tel MAS*, the astronaut may invoke the communication handler autonomy component in order to send the astronaut’s (optional telemetry) messages directly.

From a design perspective, the autonomy mode change process may be controlled through development of a coordinator/lead agent that embodies the responsibilities of the coordinator/lead role. Similarly, the requirement of each MAS to self-describe its services, etc., may be realized through instantiation of a *Description Agent* role that embodies the responsibilities to provide descriptive information, some of which is dynamic.

C. **PROTOCOL DEVELOPMENT FOR USE IN THE MULTI-AGENT SYSTEM—ASTRONAUT INTERACTION FRAMEWORK**

Protocols should be consistent with the expectations of the Mission Society. They should provide for methodical processes and support a conversation that involves

TABLE 4 ADDITIONAL REQUIREMENTS ON A MAS IN ORDER TO PARTICIPATE IN THE MISSION SOCIETY

Area of Applicability	Requirement
Functionality (for operation in different autonomy modes)	REQ F1: Each MAS must offer a design for operation in the 1) fully autonomous mode, 2) semi-autonomous mode, and 3) nonautonomous mode. (NONE will indicate null offering.)
Functionality	REQ F2: Each MAS must know its internal plan for configuring the behavior/state of its participant agents appropriate to all autonomy modes that it supports.
Functionality	REQ F3: Each MAS provides a protocol for changing from functionality of autonomy mode to functionality of another autonomy mode.
Interaction	REQ INT1: Each MAS that can operate in a nonautonomous mode must offer single designated agent to provide interface for functionality to astronaut.
Interaction	REQ INT2: Each MAS that can operate in a nonautonomous mode must provide a dialogue template to structure the interaction between the designated agent and the astronaut.
Interaction	REQ INT3: Each MAS that offers null functionality in the nonautonomous mode must support handling a communication from an astronaut to change its autonomy mode.
Interaction	REQ INT4: Each MAS that offers null functionality in the nonautonomous mode must support handling communications requesting its status or description information.
Reporting/ visualization	REQ RV1: Each MAS must provide a list, upon startup and upon any changes, of its subautonomy components and their active status, a list of its dependencies on other MAS, its current autonomy mode, its operational level, and its safety alerts to the MAS situation console for visualization.

Note that the operational status change is not the equivalent of an autonomy mode change although the consequences for users may be the same.

decision points and branching. The Mission Society also places a high premium on safety. With regard to protocol development, safety is supported through 1) nonambiguity of message and 2) interactions that support behavioral (social) expectations. *In the Mission Society, acknowledgment of communications is the expectation.* In the development of protocols for use in the Mission Society, use is made of FIPA standardized performatives and the “ack” suggested by Kremer and Flores. Selected shorter FIPA protocols were also subsumed into the Mission Society protocols that have been developed. These include the following: multi-agent system startup protocol, virtual team (Virtual MAS) startup protocol, description protocol, autonomy mode change protocol, and LRU replacement dialogue protocol. The autonomy mode change protocol is presented.

1. AUTONOMY MODE CHANGE PROTOCOL

The autonomy mode change protocol varies depending on whether the MAS is virtual, that is, whether it involves a virtual team. A virtual MAS is a team that is composed of specific infrastructure agents, such as a team lead/coordinator. However, many of the functional agents that act as part of the virtual team are those with specific expertise, which are “borrowed” from their “home” MAS.

As a result of the autonomy mode change protocol, the MAS will change the autonomy mode in which it is executing. The functionality between levels is generally different; the products and/or services offered by the MAS at different autonomy levels are not the same. *An exception to this is the description autonomy component; it responds to a “query” at all autonomy levels.*

A description of the coordinator/lead agent in coordinating and controlling the autonomy mode change is presented here with illustration of its usage in Group B (prototyped) use cases. Consider the case of the autonomy mode change protocol for a nonvirtual MAS. In the autonomy mode change protocol applicable to a nonvirtual MAS, as illustrated in Fig. 1, the coordinator/lead agent must accomplish the following tasks:

1. Inform any current customers who have a service pending (in process) of its disposition and reason. This may be *delegated* to another agent within the MAS.
2. Inform any current subscriber (to the product) customers of the change to the new autonomy mode status.
 - (a) Also inform the subscribers of their new subscription status, with the reason.
 - (b) This may be *delegated* to another agent within the MAS.
3. Inform all other agents in the MAS of the change to new autonomy mode and what activity state they will assume. In the new autonomy mode, an agent will be active or inactive,

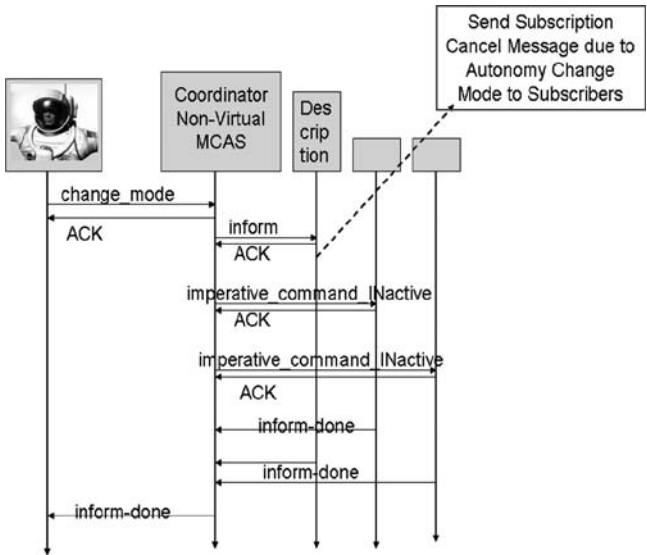


FIG. 1 Autonomy mode change protocol for a nonvirtual MAS in which only the coordinator and description autonomy components remain in active status.

4. Inform the designated agent that it should set itself to the designated agent state. Its status will be active.
5. Wait until all of the other agents have completed whatever “cleanup” they must perform in order to change over to the new mode/activity status.
6. Tell the astronaut when the autonomy mode change is completed.
7. Remain active for the astronaut to request a future autonomy mode change.

Figure 1 shows the autonomy mode change protocol applicable to a non-virtual MAS in which only the coordinator and description autonomy components (agents) remain in active status as the autonomy mode is changed. This is the case with the hardware health MAS in a change from full autonomy to nonautonomy mode.

Clearly, a change in the autonomy mode applicable to a nonvirtual MAS can ripple through and affect other MAS operations, both nonvirtual and virtual. For example, an autonomy component (agent) *may have to be* “recalled” from a virtual team. Consider the case in which a nonvirtual MAS depends on an information product that will no longer be available when the supplier MAS enters a no-autonomy mode. The dependent MAS will either not be able to offer its product/service or will offer a degraded version.

2. CASE OF THE AUTONOMY MODE CHANGE PROTOCOL FOR A VIRTUAL MAS

The autonomy mode change protocol is somewhat modified for the case of a virtual team. Some of the autonomy components (agents) that are to be designated INActive on the virtual team are those that belong to another MAS, which is non-virtual. This nonvirtual MAS must be consulted regarding proposed changes in the activity status of its agents, even though the agents in question have been “loaned out” to the virtual team MAS. See Fig. 2.

The autonomy mode change protocol from fully autonomous to no-autonomy mode is presented for the case of the power distribution device access virtual team. It is possible that the nonvirtual MAS, in this case the power system MAS, will object to having two of its team members placed on inactive status. Even though these team members have been loaned out to the virtual team, the power system MAS may have them scheduled for recall and use later.

The order to change the autonomy level originates only with the astronaut. In this case, the change in autonomy levels requires that the power access device LRU1 and LRU2 agents will be set to an inactive state. The virtual team (VT) lead/ coordinator must query for the agreement of the nonvirtual MAS regarding the future activity status of its agents. Here, the power system MAS agrees to the change.

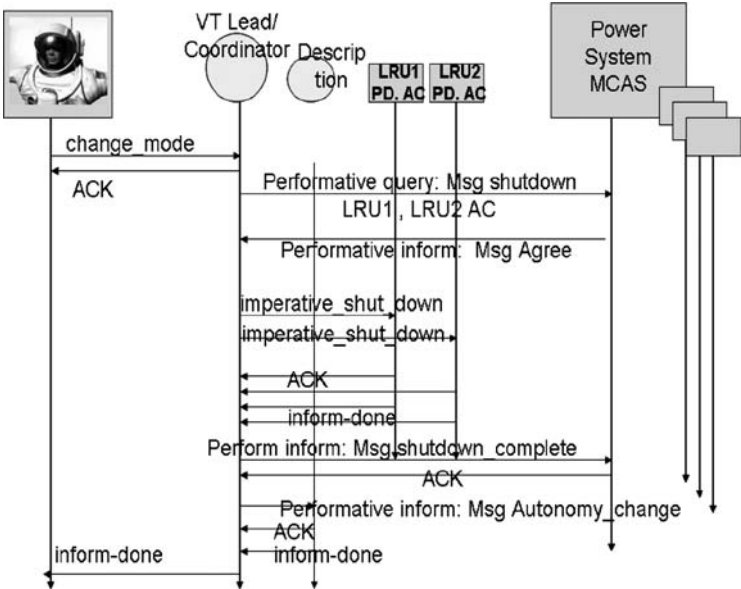


FIG. 2 Autonomy mode change protocol in virtual team; here, the nonvirtual power system MAS does *not* object to the change.

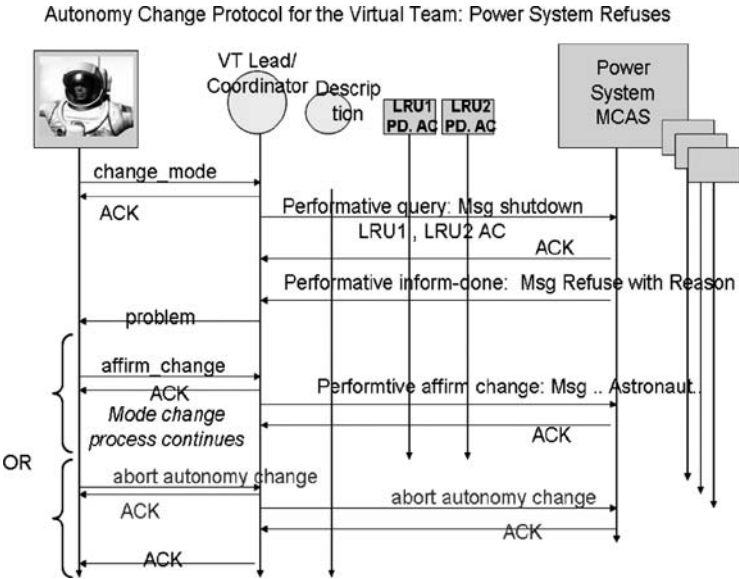


FIG. 3 Autonomy change protocol in virtual team: the nonvirtual power system MAS *does* object to the change.

The protocol unfolds with an order (imperative) to shutdown (become inactive) to the two power access device agents. They both acknowledge and perform duties to comply; at completion they provide an inform-done message to the virtual team lead/coordinator. The virtual team lead/coordinator then informs the power system MAS coordinator of the new status of two of its power access device agents. After an acknowledgment is received from the power system MAS coordinator, the virtual team lead/coordinator informs the astronaut that the autonomy mode change has been completed.

The power system MAS coordinator did not have to agree to the autonomy mode change. Perhaps such a change would impact the activity state of two of its “loaned-out” agent members, and it must disagree because these agents must remain active for future, near-term efforts. In this case, the autonomy mode change protocol would have proceeded differently. This is illustrated in Fig. 3.

When the astronaut is informed that there is a problem with the autonomy mode change, the astronaut is also informed of the reason. This reason is propagated through to the astronaut; the reason originated in the message content sent by the power system MAS coordinator. The astronaut can consider the information offered in the “problem” message. The astronaut may decide to proceed with the autonomy mode change in an “affirm_change” message, or may decide to abort the autonomy mode change. Whatever path the astronaut selects is conveyed to the power system MAS coordinator.

D. PROTOCOL UTILIZATION IN THE MULTI-AGENT SYSTEM—ASTRONAUT INTERACTION FRAMEWORK

A prototype that exposes information on the MAS applications and their execution states, as per the preceding discussions, was developed. This prototype incorporates the Mission Society elements developed for the human-multi-agent autonomous system interaction framework. The multi-agent systems of Group B use case scenarios were included.

Two consoles were developed. From the point of view of the astronaut, console 1 provides information on and situation awareness of the various multiple agent autonomy system (MAS) software applications that are extent in the spacecraft. Console 2 provides the astronaut with the ability to interact with each of these multi-agent autonomous systems. It includes capabilities that support the astronaut in ordering autonomy change, that support the astronaut engaged in an LRU dialogue, and that supports the astronaut in message sending to invoke descriptions of the status of each active MAS.

The motivation behind the development of a human–autonomous system interaction (HASI) framework is the need to expose to the astronaut information concerning all of the MAS applications that may be executing onboard the spacecraft or on mobile platforms and to provide the astronaut with a level of control over the autonomy mode in which each of the MAS applications is executing. To provide the astronaut with information, each of the MAS autonomy applications *must* provide information about itself. This is accomplished in a structured manner, with an XML schema used to describe each MAS.

All interactions that an astronaut may have with any MAS or agent in the Mission Society is through an interaction protocol, such as autonomy change, description, replacement dialogue, etc. The console interface guides the astronaut through protocol-based interactions with each MAS, as appropriate.

While the “Mission Society” may have a list of all of the MAS that are onboard, there is the possibility that an omission occurred and that an MAS is present and executing without the proper meta-data. The meta-data are captured in an associated XML schema that describes the MAS, including its dependencies. The absence of this XML document indicates that a configuration control issue has occurred. Nonetheless, the astronaut must be alerted to this MAS. The astronaut will be alerted to this, as a window on the prototype console will note the absence of an XML file. The astronaut team may then consult with mission control and reach a decision that allows the MAS to execute, etc.

Figures 4 and 5 pertain to console 1. Figure 4 is a screen shot of console 1 as it first appears. Note that console 1 offers four tabs. The front view left side provides details on the list of the nonvirtual MAS that *are known* (from configuration files) to be in the mission configuration. Each MAS shows a “tree” of information. The “autonomy mode” designation in this screen pane indicates the highest level of autonomy at which the MAS may operate. The “type” indicates a regular (nonvirtual) MAS. Products and services that a MAS may provide are listed. Each of the

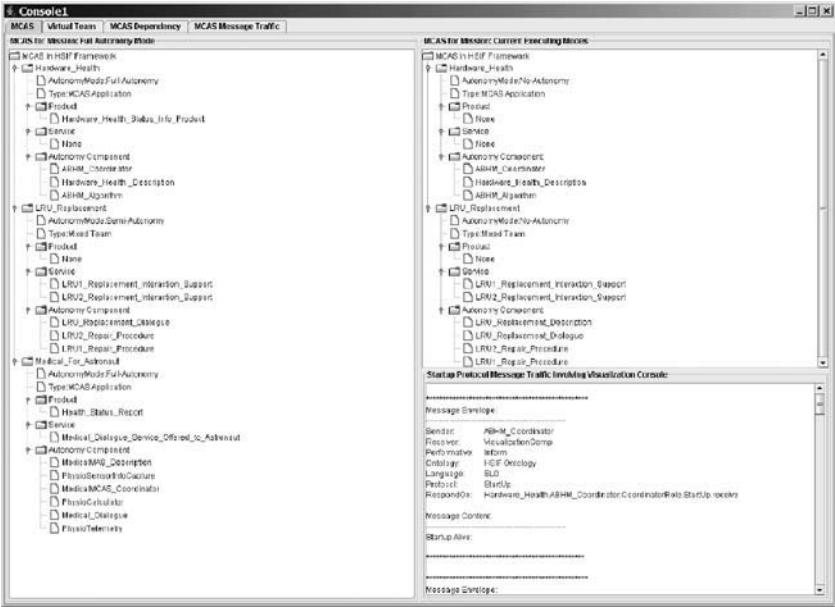


FIG. 4 Console 1, Tab 1 is an initial screen which shows nonvirtual MAS information.

autonomy components that comprise the MAS software application in its highest autonomy mode are listed individually. The front top right view shows all of the MAS that are in fact *executing* at this time. The reader may note that the “Medical for Astronaut” MAS is *not* executing as it does not appear on the execution list.

The lower right side view shows the message traffic between the coordinator autonomy component of the MAS and the console (named the “visualizer”); messages that are sent according to the startup protocol are shown in this window.

The virtual teams are shown on the second tab of console 1; the virtual team that is executing is the Power Distribution Device Access Virtual Team. The third tab offered by console 1 presents a graph view of the MAS in the mission configuration. See Fig. 5. The left-hand pane shows all MAS that are part of the mission configuration, with their offered products and services. The products and services are shown as connected to their MAS by an edge that is labeled “offers.” Dependencies of one MAS’s product or services upon those of another MAS are shown via an edge that is labeled “depends_on.” The right-hand side presents a graph of which MAS are currently executing. Note that the Medical_for_Astronaut MAS, which is not executing, appears grayed out. The astronaut is made aware of missing XML configuration information through the list of unavailable MAS in

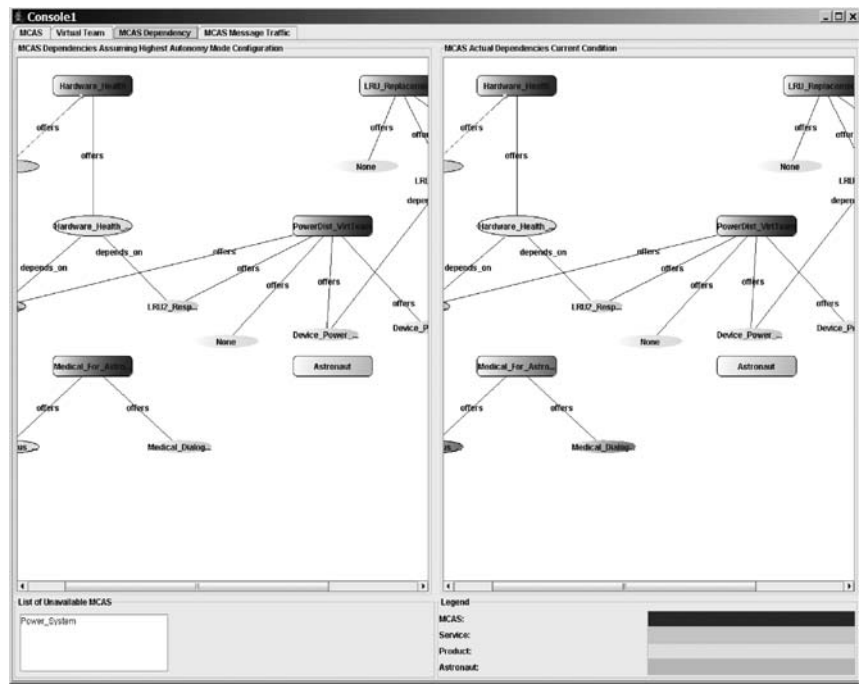


FIG. 5 Console 1 Tab 3 screen presents a graphical view of MAS, their products and services. Dependencies among MAS are shown.

the window shown on the lower left; information on this MAS was not provided to the Mission Society.

Tab 4 of console 1 provides all of the message traffic among the agents of a MAS (intra-MAS) or from an agent of one MAS to an agent of another MAS (inter-MAS). In developing autonomy applications for the mission, it is reasonable to assume that the products or services of one MAS may be accessed by another, thus entailing the need for additional message traffic. This screen view provides a window into that inter-MAS message traffic so that the astronaut may view it.

Console 2, not illustrated here, involves two tabs. Its Tab 1 provides the astronaut with dialogue support and with the ability to query the description autonomy of every MAS that is executing. The query topics that the description protocol supports are provided in a drop-down list box. Tab 2 supports the astronaut in the autonomy mode change protocol. (It also offers the ability to talk with a designated agent, provided the particular MAS has one.) For the autonomy mode change protocol, the astronaut selects an executing MAS; the autonomy mode that the MAS is operating in is shown. The astronaut selects a new

autonomy mode for that MAS; if the mode is not supported, the autonomy change protocol is not invoked, and the astronaut is informed with a message in the message box.

V. ROBOT AGENT

A. INTRODUCTION TO THE BASIC ROBOT AGENT

The underlying robotic software framework is given by the distributed control framework (DCF) [27]. Within the DCF, a robot agent is the software construct that represents the physical robot. The DCF can be viewed as a component-based architecture that utilizes an abstraction layer for the hardware elements. With DCF, the components provide the categories of activities that the robot agent must address. Other frameworks could be selected as a starting point to support the protocol-aware robot, provided the agent abstraction is support by the chosen framework.

The DCF uses the CybeleTM [28] agent development and execution environment in order to develop the agent-like aspects of the robot. These aspects are distinct from the lower-level “move” capabilities of the robot agent. As with the DCF, other agent platforms could be utilized. The CybeleTM agent is a generalized component that is composed of activities; see [29]. The specialization of the activities within these components allows the user to develop a range of capable agent types, including reactive agents, belief-desire-intent (BDI) agents, etc. The CybeleTM agent platform also provides the infrastructure needed to support multi-agent systems, include message queue management and message delivery among agents. It also supports a variety of communications modes, including TCP/IP and ad hoc network protocols in a pluggable architecture. This capability is needed by the physical, deployed robot team as well as by simulated robots that are distributed across host computers.

Figure 6 illustrates the structure of the robot agent. It is composed of activities: the state estimator activity, the coordinator activity, the planner activity and the custom task activity. From [15],

State estimation is done within the State Estimator activity; sensor input is used in estimation models; different models may be developed. The Coordinator activity is responsible for aggregating environment data. The Planning activity addresses higher level task assignment together with the lower level motion planning. This is supported by an execution engine for behaviors that are expressed in the Motion Description Language, enhanced (MDLe). [20]

The custom tasks activity allows for development of tasks that are specific to the particular application.

The MDLe allows the designer to develop more complex movement sequences from individual “atoms.” Details on how this is done are provided in [30]. Within MDL a kinematic state machine (KSM) formalism is utilized. Control laws, which

are applied to the physical plant, serve to form the individual state. Transition from state to state is driven by interrupts.

B. DEVELOPING THE “SMARTER” ROBOT AGENT

The starting point for development of a cognitively enhanced robot agent that will support protocol utilization is taken to be the robot agent as defined in the distributed control framework. Within that framework, the robot agent is already capable of receiving and sending message transmissions. The enhancement that is needed is the ability to recognize the protocol-bound messages that are utilized in the Mission Society, to extract information encapsulated in the messages and to utilize the information in decision making. Additionally, each robot agent must be able to respond to other robot agents in a protocol-aware manner. To support these needs, a modular manager structure and a decision-making structure were designed and developed.

The development proceeded within the context of robot agents that assumed either a worker or team leader role. To further ground the development, details are developed in terms of a notional scientific sampling mission. Detail on the mission is given in following sections.

1. MANAGER STRUCTURE

The manager structure is hosted in the planner area of the robot agent; this is indicated by the circle around the term “planners” in Fig. 6. The cognitive enhancement to the robot agent is provided by the three-element manager structure. Three manager modules have been developed: 1) conversation manager, 2)

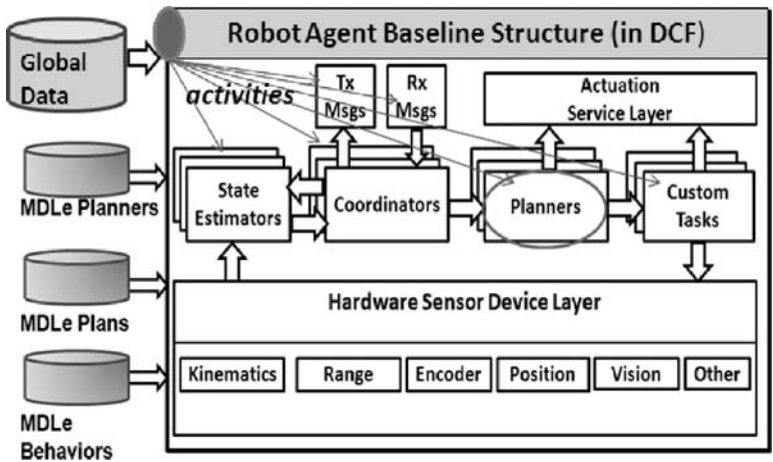


FIG. 6 Robot agent within the distributed control framework structure (adapted from [27]).

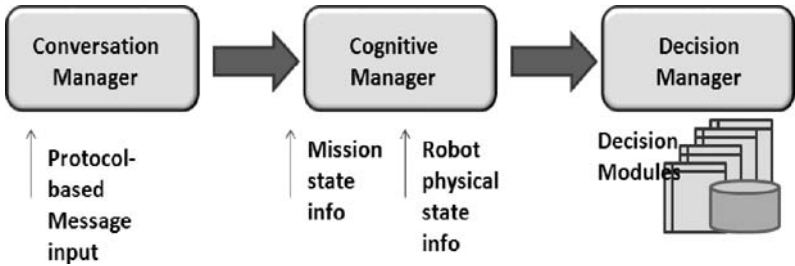


FIG. 7 Robot agent’s initial lower-level cognitive capability is replaced by a modular manager structure.

cognitive manager, and the 3) decision manager. Each of these managers is invoked in sequential order within the robot agent. Figure 7 illustrates this; the invocation is from left to right.

Conversation manager. The conversation manager has several tasks. First, it must process the protocol aspects of incoming messages. Concurrent conversations may be occurring; the state of each conversation vis à vis the governing protocol must be maintained. The message content is extracted from each message and placed in a data store that is accessed by the cognitive manager. The final responsibility of the conversation manager is to provide the decision manager with the next (reply) step in the protocol-based message exchange for each conversation for those decisions that involve the action of message generation and sending.

Cognitive manager. The cognitive manager receives inputs from the lower level of the DCF framework, providing sensor results. The DCF structure processes these and provides a map that the cognitive manager accesses. This provides information on the robot-self state. The cognitive manager, with access to the message content extracted by the conversation manager, must extract information content from the messages. Such content would typically contain mission status information. Typically, these messages would be highly structured as per the mission needs. Additional information regarding robot team member or astronaut interaction updates is also captured by the conversation manager. The conversation manager develops the knowledge base reflecting the mission status and the self-status of the robot agent. It is this knowledge base that is used by the robot agent’s decision manager.

The concept of “status dimensions” is introduced so as to provide a means of organizing information. A state is then specified as the settings (values) over a set of status dimensions. Table 5, introduced in [15], shows the status dimensions that are applicable to both team leader and worker robot agents. A mission other than the scientific sampling mission may involve different or additional status dimensions. Within the context of the notional scientific sampling mission, the

TABLE 5 STATUS DIMENSIONS FOR BOTH TEAM LEADER AND WORKER ROBOTS [15]

Status Dimension	Status Dimension
Robot_self-status	Mission_operating_status
Proximity_status	Team_effort_status
Safing_effort_status	Problem_status
Recharging_effort_status	

team leader robot agent also has a mission_load status dimension. We note that that status dimensions map to a focus that is organized by protocol type. However, the mission needs also demand utilization of such protocol types.

Decision manager. The decision manager is responsible for selecting the next action or actions that the robot agent takes. Types of actions the decision manager can take to support the mission and safety objectives include 1) message sending, as per the protocol structure, and 2) physical actions made possible by actuators.

2. DECISION-MAKING STRUCTURES

The cognitive structure in the decision manager uses a set of complex rules to determine the next actions. As presented in [15], each rule may be written as

{State_1, State_2, . . . } => DecisionConditions := Action Selection

Complexity arises due to the multiple settings (values) that are possible for the status information dimensions. The decision conditions, which form the decision rules, may involve the use of settings. A finite state machine representation for each decision foci was developed as a precursor to encoding the decision conditions [26]. A state within the FSM is defined as the union of (relevant) status

TABLE 6 ALLOWABLE VALUES OF SELECTED STATE DIMENSIONS FOR WORKER ROBOT AGENTS (IN THE SCIENTIFIC SAMPLING MISSION [15])

Operating Status Mode	Proximity Status	Problem Report Status
WORKING	ASTRONAUT_NEAR	UPDATE_REPORTED
SAFED	NONE	UPDATE_NOT_REPORTED
WAITING	---	OK
RECHARGING	---	---
NON-OPERATIONAL	---	---

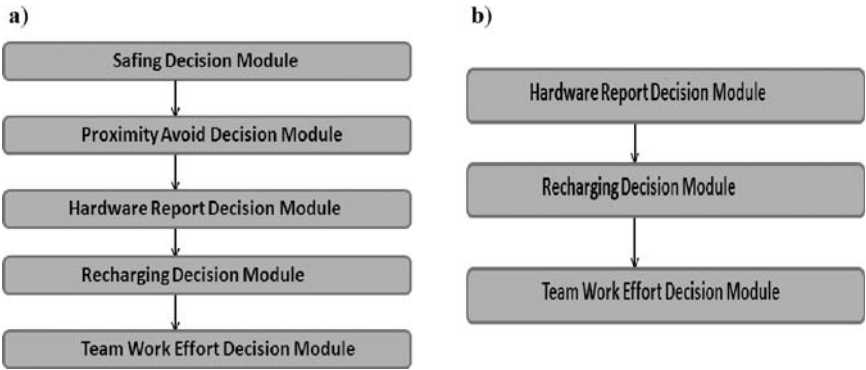


FIG. 8 Decision modules in team leader robot agent’s decision manager a) for a complex mission where safety is paramount and b) relevant to a mission with the reduced mission parameters.

dimension variables with a specific value. Table 6, presented in [15], provides allowable values for several status dimensions.

As noted earlier, the protocol-related foci can serve to organize rules in modules. A decision manager must process each module of concerns; it executes the rule sets in the order of each of the modules. As noted in [15], the order of the modules is “flexible and reflects the priorities of the domain.” However, not all mission instances must have the same set of concerns. For example, a mission involving only robot agent and having each robot dispersed to a nonoverlapping area and remaining in its area does not have need for proximity avoid or safing decision modules. Figure 8a presents a set of decision modules for the team leader robot agent, in execution order, for a mission involving astronauts and robot agents (with team leader and worker robots) as well as possible overlapping areas or routes of travel. Figure 8a is a revised, updated version of that presented in [15]. Figure 8b presents a set of decision modules for team leader robot agent in a mission having much reduced mission parameters.

VI. AUTONOMOUS ROBOT TEAMS AND ASTRONAUT OVERSIGHT: PROTOCOL DEVELOPMENT

A. MOTIVATING SCENARIO

A notional, abstract lunar environment forms the context of the notional scientific sampling mission. An autonomous team of robots has been assigned to Area 7 in support of the mission. The assumption is that Area 7 has been presurveyed and the rock sampling areas delineated beforehand. A communications grid is assumed to be in place such that the robots can interact with each other as well

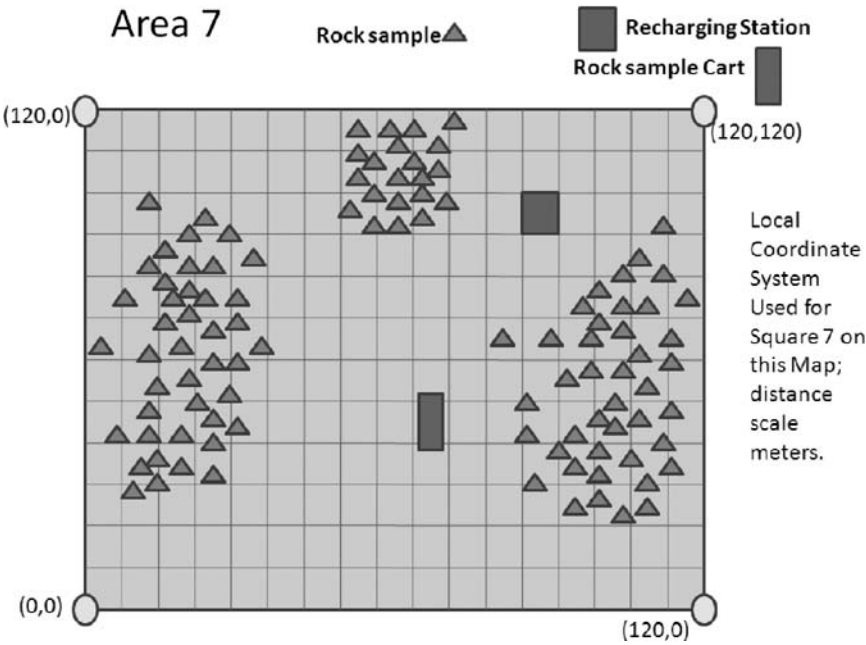


FIG. 9 Notional lunar area for scenario execution.

as receive messages originating from the lunar outpost or from a specific astronaut. Note that the focus is *not* on the communications infrastructure. However, protocol design and the enhanced cognitive structures in the robot agent offer robustness against dropped messages.

The overall scenario activity is that of the autonomous team work effort; worker robots are participating in a teamwork effort under the direction of the team leader robot. The environment of Area 7 is shown in Fig. 9. The sample card is denoted by the rectangle. The square denotes a recharging station. Rock samples themselves are indicated by triangles.

As participants in the scientific sampling mission, each worker robot will inspect an individual rock. Rocks with certain characteristics are samples of interest; these must be placed in the rock cart. In the simulation, the result of the robot's inspection of the sample is probabilistically determined. If a sample is found to be of scientific interest, the worker robot must leave its area of activity and carry the rock sample to the cart, place it in the cart, and return to its area of sampling activity. In the scenario, the team leader robot is assumed to remain at or near its initial deployed location.

At any time during the scientific sampling mission, various issues or situations may arise. An astronaut may walk through the work area as part of an inspection. In response to this, the worker robots and the team leader robot should adjust

their behavior and communicate their intentions via the use of the safety protocols (to be discussed). A worker robot may find that it is running low on battery power; the battery must be recharged. A worker robot could also suffer damage to part of its sampling equipment or find its mobility hindered. The team leader robot could receive a message from an astronaut or from an entity (software agent or astronaut) at the lunar outpost telling it to end the mission. Communications among the members of the autonomous robot team must convey relevant information in these situations. Relevant protocols, presented in the subsequent section, are designed to support needed robot interactions.

B. PROTOCOL DEVELOPMENT

A basic assumption regarding the protocol development for the autonomous robot team engaged in the scientific sampling mission is that the teamwork effort is central to accomplishing mission tasks; see Fig. 10. It is this activity that should be supported by protocol. Although a teamwork effort protocol was developed, this was not the only protocol needed to support the broader robot interaction needs, their behavior in the presence of an astronaut, or the protocol-supported actions that an astronaut could take vis à vis the robot team.

The teamwork effort (TWE) protocol concept provided a *starting point* for the protocol development effort. Initially, TWE was a larger protocol that was less focused in the sense that it incorporated interaction segments that were ancillary to the team work effort. For example, the problem reporting protocol was initially considered as part of TWE. As part of the iterative process of protocol design, protocol development ultimately distinguished between the core teamwork effort needs and supporting needs, such as problem reporting, recharging,

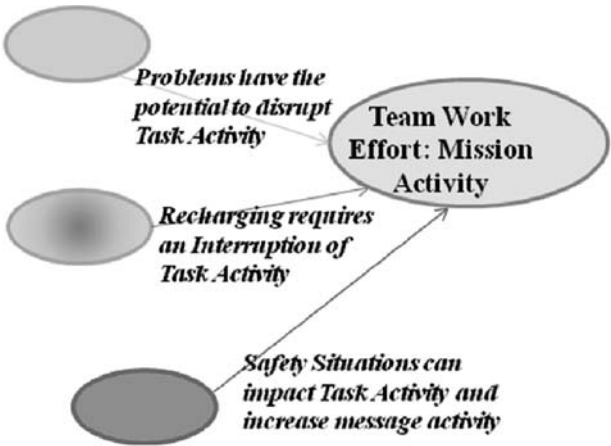


FIG. 10 TWE protocol is central protocol for mission.

and safety. Table 7 indicates the major decisions taken in the iterative process and the rationale for them.

Tables 8 and 9 list the protocols that have been developed as part of the effort. They are separated into two groups: the first in which the robots behave as autonomous and active entities and the second in which the astronaut directly addresses a specific robot.

The more complex protocols developed to support the scientific sampling mission are the teamwork effort protocol and the recharging protocol. Each of these involves phases (states) within the protocol. A protocol phase may be defined as a shorter segment embedded in a larger, more complex protocol. The more complex protocol supports an activity that is contributing to an action intent that is the reason for the complex protocol. It does this through

TABLE 7 FEATURES OF THE PROTOCOL DEVELOPMENT ITERATIVE PROCESS

Decision	Rationale
Refactor the initial teamwork effort protocol	Separate the core teamwork effort needs from important, but ancillary, concerns such as safety, battery recharge needs and problem reporting.
Decomposition: introduce the notion of “phase” within a protocol	Decomposition of each protocol was considered. Certain physical and task-related processes have subareas of concern; the phase notion reflects this.
Additional information inclusion: notion of context within the header (envelope information) associated with each performative	Introduce the notion of a protocol <i>context</i> to provide more protocol-related information. This was found to be helpful in managing the conversations. This is instantiated as part of the message header.
Role: further specification of the general activity scope within the scientific sampling mission that an agent (human or robot) has	As part of the protocol decomposition and structuring, it was found that certain roles would consistently act as initiators or recipients in specific protocol-managed conversation segments. The <i>role</i> is assigned to an agent to indicate its part in the scientific sampling mission. Roles include astronaut_human, worker_robot, teamleader_robot. Role information is used to manage the conversational message exchanges.
Use of imperative for performative	The “directive” is used by the team leader as it coordinates the major phases of activity of the workers within the greater mission.

TABLE 8 PROTOCOLS INVOLVING ROBOT-TO-ROBOT TEAM MESSAGING ACTIVITY

Protocol Name	Comments
Teamwork effort protocol	Multiple worker robot agents and a team leader robot agent engage in a team effort using this protocol.
Protocol for astronaut oversight of the teamwork effort	The astronaut provides oversight at specific junctures in the execution of the teamwork effort protocol.
Problem protocol	A worker robot reports a protocol with its hardware or software using this protocol.
Recharging protocol	A worker robot that needs to recharge at a station located in an area that has much mission activity by other robots will utilize this protocol.
Safety-related protocols: robot initiated	There are three such protocols: 1) Proximity safing (robot self-safing, in presence of an astronaut) 2) Proximity avoid (avoid a region) 3) Proximity alert (moving/ traveling through an area)

the message flows that are organized; part of the message flows are found in the shorter phase (state) segments.
As stated in [15],

messages that involve coordination point issues or that are final messages (from one of the communicants) are *always acknowledged* by the receiver, consistent with Mission Society norms. The acknowledgment may be an explicit “ACK” message. Alternatively, an acknowledgment may be in the form of a semantically appropriate response that is governed by shorter protocol segment. For example, a “query” message from Robot A may be answered by an “inform” message, sent from Robot B. The “query → inform” sequence has been standardized by FIPA. Both “query” and “inform” indicate the *action* of the message; these actions, termed *performatives*, are consistent with the message content.

TABLE 9 PROTOCOLS INVOLVING ASTRONAUT INITIATED MESSAGE EXCHANGE WITH A SPECIFIC ROBOT

Protocol Name	Comments
External safing or desafing	An astronaut orders a specific robot to safe or desafe.
Astronaut check status	An astronaut orders a specific robot to report its status.
Astronaut high level supervision	The astronaut supervises the team leader at key steps in its management of the teamwork effort (protocol).

1. TEAMWORK EFFORT PROTOCOL

The TWE supports mission-oriented teamwork among autonomous robots. Each of the robots, represented by a robot agent, is assumed to have a specific role in the teamwork effort. In this work, the roles of *team leader* and *worker* were defined. The TWE protocol defines allowable message types, governing the message exchanges between a team leader robot agent and the worker robot agents. The TWE protocol is composed of five states, corresponding to five potential mission phases. These are the mission: 1) initiation, 2) execution, 3) interruption, 4) resumption, and 5) termination.

Each protocol phase utilizes shorter, relevant protocols, defining allowable message exchange sequences, which advance or further structure the mission activities. A diverse set of mission sequences is permitted within the TWE protocol. These include sequences such as the following:

1. Mission Initiation, Mission Execution, Mission Termination;
2. Mission Initiation, Mission Execution, Mission Interruption, Mission Resumption, Mission Termination; and
3. Mission Initiation, Mission Execution, Mission Interruption, Mission Termination.

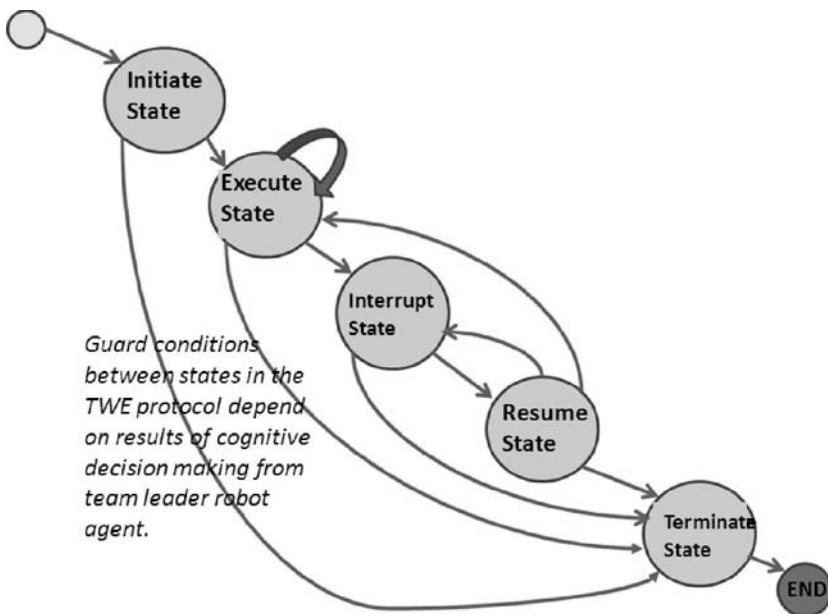


FIG. 11 Finite state machine view of the TWE protocol (adapted from [15]).

TABLE 10 SIMPLER PROTOCOLS THAT MAY BE UTILIZED IN THE EXECUTION STATE OF THE TWE PROTOCOL

Simpler Protocols	Description
Report status	Worker robot agent proactively provides its status.
Query status	Team leader robot agent asks a worker robot agent for its status.
Change task	Team leader robot agent modifies a task of a worker robot agent.

The shortest sequence is noted first. It proceeds on a linear path, from mission initiation to execution to termination. This is from the TWE protocol perspective, which does not convey information as to mission success. For example, a mission may have been terminated early due to external reasons, leaving only a fraction of the intended rock assay completed. The second and third sequences listed indicate mission interruption, while the third includes a mission resumption phase. The allowable sequences are those that satisfy allowed transitions to specified states in the TWE protocol finite state machine that is shown in Fig. 11.

Note that the initiation phase occurs only once. The termination phase, once entered, leads to the end of the interactions governed by the TWE protocol. The termination phase may be entered from all other phases. The team leader robot agent is in charge of the transition from/into each state (phase). The results of the cognitive decision making rules form the guard conditions for transition. However, the input into these rules may originate with information from the worker robot agents, from astronaut input, from input by planning agents at the lunar outpost, etc. Additional details are found in [15].

Other protocols may be utilized within the scope of the TWE protocol. For example, the execution state of the TWE protocol may involve “smaller” conversations that are based on other relevant protocols. Table 10 provides a list of the subphase protocols within the TWE protocol’s execution phase.

2. TWE PROTOCOL MODIFICATIONS TO ALLOW ASTRONAUT OVERSIGHT OF TEAMWORK EFFORT

Astronauts may wish to have a greater level of oversight over the autonomous robots’ teamwork efforts. The state of the current practice [8] allows an astronaut to interact with a robot directly at a fine-grained level of control vis à vis the tasks. In our approach, the astronaut interacts *only* with the team leader robot. Rather than having an astronaut micromanage a worker robot, the astronaut exercises oversight by granting or denying the team leader robot permission to proceed with the teamwork effort at certain coordination points, as defined by an extended protocol named “Astronaut Oversight of the Teamwork Effort.” It allows astronaut oversight in three phases: 1) oversight for starting initiation, 2) oversight for starting execution, and 3) oversight for termination. Table 11, an extension

**TABLE 11 ASTRONAUT OVERSIGHT OF TEAM EFFORT PROTOCOL: TERMINATE PHASE,
WITH REQUEST INITIATED BY THE TEAM LEADER ROBOT**

Worker Robot Agent	Team Leader Robot Agent	Message	Astronaut
	X → (sends)	Permission query to Proceed to Terminate	X
		ACK	← X (sends)
		Query: status	← X
	X →	Reply on status	
		Directive: Proceed_Terminate or Deny_Terminate : Assume the case is: OK_Proceed _ Terminate	
	X	OK_Proceed_Terminate	← X
	X →	ACK	X
X	<- X (sends)	Transition to termination (to each worker robot agent)	
X -> (sends)		ACK	
X -> (sends)		Inform_done	
	X →	Inform_done (Terminate ended) [after all worker robot agents report)	X
	X	ACK	← X

of [15] to include the participation of the worker robot agents, illustrates the message exchanges that take place for the terminate state/phase.

Worker robot agents never interact with the astronaut regarding the tasks activity. Rather, the team leader robot agent coordinates with them as per the teamwork effort protocol, gathering enough information to allow it to interact with the astronaut in his/her oversight of the teamwork effort. Worker robots have no knowledge that the team leader robot agent and the astronaut are engaged in a protocol-compliant conversation regarding when phases of the teamwork effort might be started.

Either the team leader robot agent or the astronaut can assume the initiative in seeking to terminate the effort. Table 11 provides the process for the case in which the team leader robot is the initiator. The astronaut responds initially responds by seeking more information, using a status query interaction. The team leader robot,

in this case, is assumed to know, before seeking permission, the mission status. Otherwise, it would have to query the various worker robot agents for input. Note that the “inform_done” from the team leader to the astronaut indicates *that the termination phase has been completed*; this may only be sent after all of the worker robot agents have sent the team leader robot agent their “inform_done” responses indicating each has terminated. Their termination can involve physical configuration changes that take some time to complete.

3. RECHARGING PROTOCOL

The recharging protocol supports the interactions between a team leader robot and the worker robot that support battery recharging. The recharging protocol is broken into three states or phases utilized sequentially: 1) setup (S), 2) recharging (R), and 3) post_recharge (PR).

Tables 12 details the message exchanges, the sender and receiver types, and the performative and contexts of the recharging protocol details *for the case in which all proceeds normally*. At the end of the setup phase, the worker robot has acknowledged an order (directive) to proceed to the location at which the charging station is located. The recharging phase starts when the worker robot informs the team leader that it has reached the charging station location. The team leader acknowledges this. The worker robot then recharges and begins charging. The worker robot will inform the team leader that it has finished its recharging with a message bearing an “inform_done” performative. Note that the worker robot has the initiative in this phase. Also note that the team leader does not have the capability to cancel the recharging process once it has initially started it (via the setup phase). It is possible that the team leader should be afforded the option to cancel in the field.

After the recharging phase is finished, the team leader robot once again has the initiative. The worker robot remains at the recharging station’s location until the team leader sends it a message to proceed to a specified location. The specified location does not have to be back to the original task area. During the time that the worker robot was being recharged, that task area may have been addressed by another worker robot. Also, mission parameters may have changed. For example, the mission commander could have decided that the number of sample rocks collected were sufficient. Alternatively, the rock cart could be full. The other worker robots might be idled, waiting for a visit from another robot team to bring a new rock cart (and possibly return with the full one). This structure of the recharging protocol supports flexibility in the robot team activities.

4. ROBOT-INITIATED SAFETY PROTOCOL: PROXIMITY SAFING PROTOCOL

The proximity safing protocol is a robot’s self-safing capability, which is used in situations in which an astronaut is in the area and crosses into a “too-close”

TABLE 12 EXAMPLE OF ONE THREAD OF MESSAGE EXCHANGES SUPPORTED IN DIFFERENT PHASES OF RECHARGING PROTOCOL

Phase	Msg in Phase	Sender	Receiver	Performative	Context
S	1	Team leader robot	Worker robot needing a recharge	Inform	RECHARGE
S	2	Worker robot needing a recharge	Team leader robot	ACK	RECHARGE
S	3	Team leader robot	Worker robot needing a recharge	DIRECTIVE	PROCEED_TO_LOCATION
S	4	Worker robot needing a recharge	Team leader robot	ACK	PROCEED_TO_LOCATION
R	1	Worker robot needing a recharge	Team leader robot	INFORM	CHARGING_STATION
R	2	Team leader robot	Worker robot needing a recharge	ACK	CHARGING_STATION
R	3	Worker robot needing a recharge	Team leader robot	INFORM_DONE	CHARGING_STATION
PR	1	Team leader robot	Worker robot just recharged	DIRECTIVE	PROCEED_TO_LOCATION
PR	2	Worker robot just recharged	Team leader robot	ACK	PROCEED_TO_LOCATION
PR	3	Worker robot just recharged	Team leader robot	INFORM_DONE	WAITING_IN_PLACE

zone around *any* of the robots, be it a worker robot or a team leader robot. *This protocol supports unconditional safety for the astronaut.*

As soon as any robot is aware of an astronaut nearby, it sends out a message having a performative “warning” and a context of “danger.” The message is sent out while it is in the process of safing itself. Because the safing process can involve hardware deployment or configuration changes (in the general case), this first message is sent out. The recipients are the astronaut and any other robots in the area.

Note that the value of “too-close” or “nearby” is determined by the particulars of the robots and the environment. After the robot has safed itself, it sends out a second message, which has performative “warning” and a context of “safed.” This message is also sent out to the astronaut and any other robots in the area. If the astronaut moves through the area that is close to the robot, eventually it will not be “too-close” or “nearby.” The robot does not have to stay in a safe mode; it can return to performing its tasks. The robot then starts the process of “de-safing” and sends out a third message, which has performative “warning” and a context of “desafed.” This message is sent out to the astronaut and any other robots in the area.

Figure 12 provides a schematic of the proximity safing protocol. The direction of the arrows indicates the initiating speaker; in this case it is worker robot 1. Note

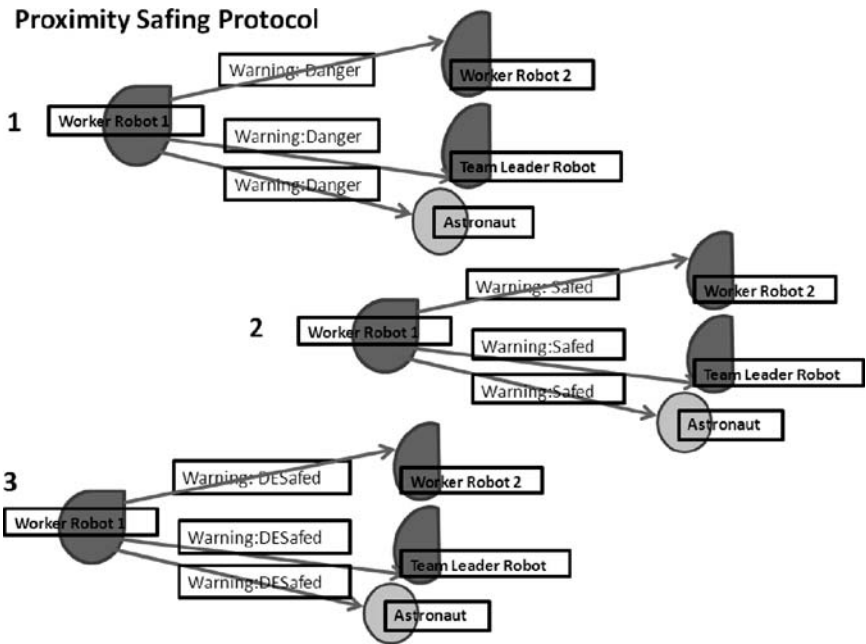


FIG. 12 Schematic illustrating states of the proximity safety protocol. Worker robot 1 is self-safing and then desafing after the astronaut has left the nearby area.

that the sender in all of the messages is the initiating robot. Worker robot 2, the team leader robot, and the astronaut are the recipients.

5. ROBOT-INITIATED SAFETY PROTOCOL: PROXIMITY AVOID PROTOCOL

The proximity avoid protocol is applicable in situations in which a worker robot is moving about within its assigned work/task area. The worker robot would like to work as efficiently as possible performing its tasks. If it is the sole worker robot in its assigned task area, then it sends out a “claim” to that area while it is working.

With respect to the scientific sampling mission scenario, each worker robot has an assigned area in which to handle and analyze rocks. If a rock is found suitable, the worker robot carries the rock over to a cart; this entails the worker robot leaving its task area. The worker robot cancels its claim to the work area when it leaves it en route to the rock cart (see Fig. 13). Other robot agents can use this information to plan a path that intersects the “claimed” area as little as possible. Robot collisions become less probable.

6. ROBOT-INITIATED SAFETY PROTOCOL: PROXIMITY ALERT PROTOCOL

This protocol is used in situations in which a robot is “on the move” out of the area that it is most commonly working in and which it has “claimed.” This is a single message protocol. The message alerts other autonomous entities, whether they are

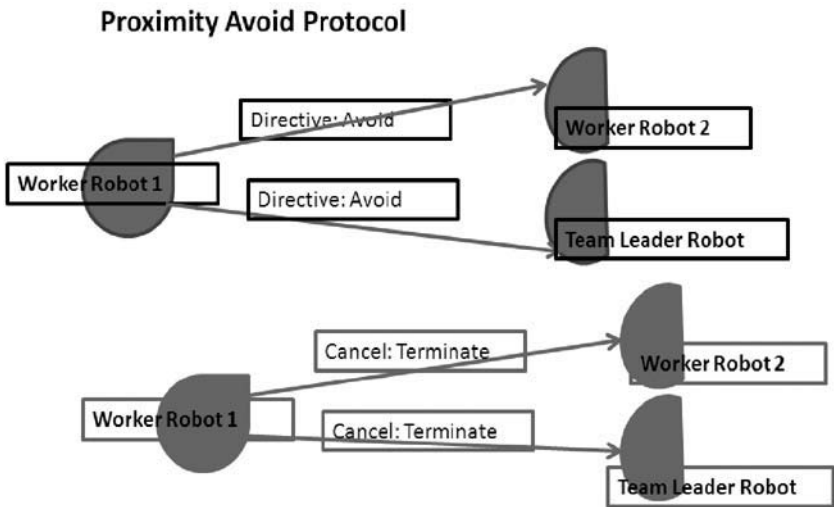


FIG. 13 Worker robot 1 is working in a specific area and tells others to avoid; this worker can cancel the “avoid” message.

TABLE 13 DETAILS OF THE PROXIMITY ALERT PROTOCOL

Message Exchange	Sender	Receiver	Performative	Context
1	The specific robot that is “on the move”	All entities in the area (other robots and astronaut)	Inform	Danger

robots, softbots, or astronauts, that a robot is transiting through an area. Information is included in the body of the proximity alert message as to the moving robot’s position and direction of movement.

Clearly, this protocol supports the safety of the autonomous teamwork effort protocol; this can be utilized by a worker robot as it moves to deposit its sample into the rock cart. However, the proximity alert protocol should be used by any autonomous, mobile entity, as it transits a region. For example, an astronaut can announce his or her position and direction of motion by using this protocol. The frequency with which the proximity protocol should be used by a moving autonomous entity depends on the speed at which it is moving, the reliability of communications, the entity density in the area, and the cost of communications.

The performative and context for each step in the message exchange set of this protocol are specified in Table 13.

C. PROTOCOL UTILIZATION IN SIMULATED AUTONOMOUS ROBOT TEAM EFFORT: SAFETY FOCUS

Figures 14–16 provide simulation results that show the team worker robots utilizing the safety protocols while carrying out their team efforts on the scientific sampling mission. Figure 14 shows a worker robot, having arrived in its work area, as per its efforts under the teamwork effort protocol. The worker robot has arrived at the work area while executing the mission; the worker robot then notifies all other robots and astronauts to *avoid* this active work area. The notification is made using the proximity avoid protocol. This area is indicated by the circle at the bottom of Fig. 14. The Active Conversations console shows that a proximity avoid protocol message was sent by Worker Robot_001 (at the indicated time) and received by the Worker Robot_000, the Team Leader Robot_000, and the astronaut. While executing the scientific sampling mission (of the To do that, the Worker Robot_001 must leave the work area; this necessitates messages to any other robots or astronauts in the area. Worker Robot_001 *cancels* the proximity alert, informing all other entities in the area that it is no longer actively working the specified area. Each conversation is tagged as completed; the receiving agents no longer carry that protocol-driven conversation as an active conversation. When Worker Robot_001 later returns to its work area after having

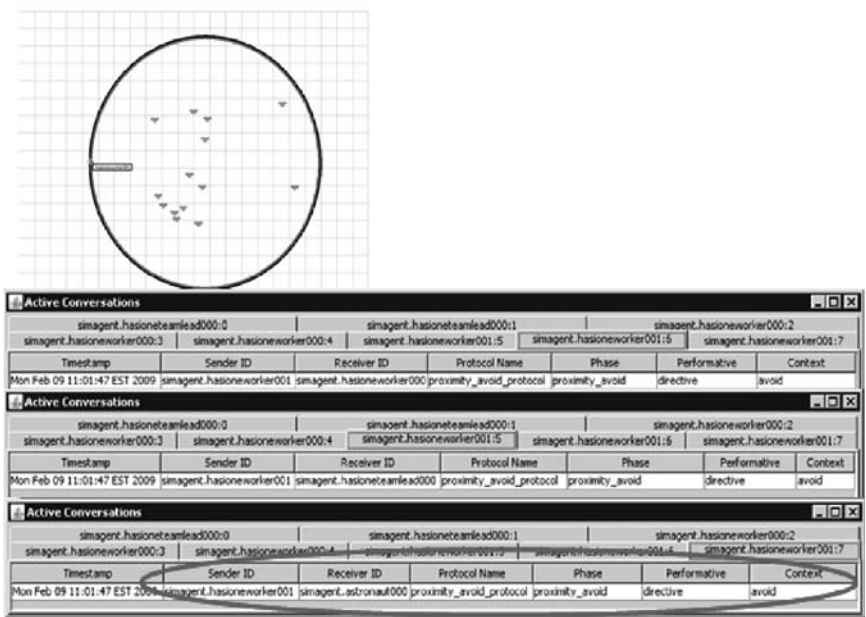


FIG. 14 Worker Robot_001, in its rock-sampling work area, issues a proximity avoid claim on the work area.

deposited the sample in the rock cart, it must start a new Proximity-Avoid driven conversation (see Fig. 15).

As the simulation progresses, an astronaut is walking towards the rock cart. The astronaut's exclusion radius is indicated by a circle around him or her. This exclusion radius is the minimum allowable distance that an astronaut may be from any robot before that robot proactively safes itself. The robot will not proactively desafe itself until after the astronaut has passed beyond the exclusion radius. From a deployment perspective, the exclusion radius must be large enough to allow the safing process to be complete before the astronaut is too close; thus, the radius is set with to be the sum of {true allowable distance} plus {distance that astronaut can walk in time that robot requires for safing}. Figure 16 shows the astronaut walking towards the top area in this view, which is at the level of rock cart, indicated in the figure. Note that both worker robots are presently depositing samples at the cart. As the astronaut is walking, the team leader is now within the exclusion zone of the astronaut; this is indicated by the aqua blue circle.

The evolving simulation is not deterministic. It must be emphasize that there is no script driving the simulation. Worker robot activity is initiated by the team

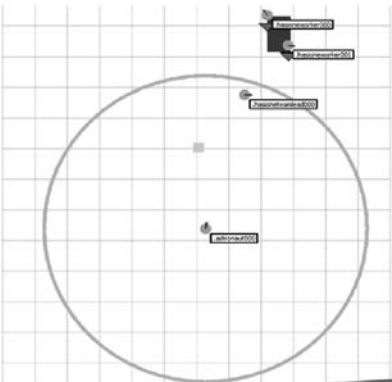
Active Conversations							
simagent.hasoneteamlead000:0		simagent.hasoneteamlead000:1		COMPLETED:simagent.hasoneworker000:2		COMPLETED:simagent.hasoneworker000:3	
COMPLETED:simagent.hasoneworker000:4		COMPLETED:simagent.hasoneworker001:5		COMPLETED:simagent.hasoneworker001:6		COMPLETED:simagent.hasoneworker001:7	
Timestamp	Sender ID	Receiver ID	Protocol Name	Phase	Performative	Context	
Mon Feb 09 11:01:47 EST 2009	simagent.hasoneworker001	simagent.hasoneteamlead000	proximity_avoid_protocol	proximity_avoid	directive	avoid	
Mon Feb 09 11:17:12 EST 2009	simagent.hasoneworker001	simagent.hasoneteamlead000	proximity_avoid_protocol	proximity_avoid	cancel	terminate	

Active Conversations							
simagent.hasoneteamlead000:0		simagent.hasoneteamlead000:1		COMPLETED:simagent.hasoneworker000:2		COMPLETED:simagent.hasoneworker000:3	
COMPLETED:simagent.hasoneworker000:4		COMPLETED:simagent.hasoneworker001:5		COMPLETED:simagent.hasoneworker001:6		COMPLETED:simagent.hasoneworker001:7	
Timestamp	Sender ID	Receiver ID	Protocol Name	Phase	Performative	Context	
Mon Feb 09 11:01:47 EST 2009	simagent.hasoneworker001	simagent.hasoneworker000	proximity_avoid_protocol	proximity_avoid	directive	avoid	
Mon Feb 09 11:17:12 EST 2009	simagent.hasoneworker001	simagent.hasoneworker000	proximity_avoid_protocol	proximity_avoid	cancel	terminate	

Active Conversations							
simagent.hasoneteamlead000:0		simagent.hasoneteamlead000:1		COMPLETED:simagent.hasoneworker000:2		COMPLETED:simagent.hasoneworker000:3	
COMPLETED:simagent.hasoneworker000:4		COMPLETED:simagent.hasoneworker001:5		COMPLETED:simagent.hasoneworker001:6		COMPLETED:simagent.hasoneworker001:7	
Timestamp	Sender ID	Receiver ID	Protocol Name	Phase	Performative	Context	
Mon Feb 09 11:01:47 EST 2009	simagent.hasoneworker001	simagent.hasoneteamlead000	proximity_avoid_protocol	proximity_avoid	directive	avoid	
Mon Feb 09 11:17:12 EST 2009	simagent.hasoneworker001	simagent.hasoneteamlead000	proximity_avoid_protocol	proximity_avoid	cancel	terminate	

FIG. 15 Worker Robot_001, upon leaving its rock-sampling work area, issues a cancellation of it earlier proximity avoid directive.

lead robot, as part of the teamwork effort protocol. The details of the activity are in keeping with the mission plan; plan information indicates the work areas for potential sample collection. The robots utilize the safety protocols when the circumstances warrant. Sample evaluation by a robot is nondeterministic. Because it is the result of each sample evaluation that drives when a robot will



Timestamp	Sender ID	Receiver ID	Protocol Name	Phase	Performative	Context	
Mon Feb 09 15:26:47 EST 2009	simagent.hasoneteamlead000	simagent.astronaut000	proximity_safing_protocol	proximity_safing	warning	danger	
Mon Feb 09 15:26:47 EST 2009	simagent.hasoneteamlead000	simagent.astronaut000	proximity_safing_protocol	proximity_safing	warning	safed	

Timestamp	Sender ID	Receiver ID	Protocol Name	Phase	Performative	Context	
Mon Feb 09 15:26:47 EST 2009	simagent.hasoneteamlead000	simagent.hasoneworker000	proximity_safing_protocol	proximity_safing	warning	danger	
Mon Feb 09 15:26:47 EST 2009	simagent.hasoneteamlead000	simagent.hasoneworker000	proximity_safing_protocol	proximity_safing	warning	safed	

Timestamp	Sender ID	Receiver ID	Protocol Name	Phase	Performative	Context	
Mon Feb 09 15:26:47 EST 2009	simagent.hasoneteamlead000	simagent.hasoneworker001	proximity_safing_protocol	proximity_safing	warning	danger	
Mon Feb 09 15:26:47 EST 2009	simagent.hasoneteamlead000	simagent.hasoneworker001	proximity_safing_protocol	proximity_safing	warning	safed	

FIG. 16 As the astronaut's exclusion zone encompasses the team leader robot, the team leader safes itself and informs other robots in the area.

leave its work area in order to place a sample in the rock cart, the motion of the robot within Area 7 of the notional sampling mission is also nondeterministic.

Simulation results are that 1) the worker robots canceled their “claim” to an area each time they left the area to place a sample in the rock cart, 2) the worker robots initiated a proximity avoid protocol when transiting to the rock cart and back to their respective work areas, and 3) all robots proactively safed themselves when an astronaut entered the edge of the exclusion zone relative to the particular robot. Multiple simulation runs were done. The protocols were judged to be working properly and in a timely manner.

D. PROTOCOL UTILIZATION IN SIMULATED AUTONOMOUS ROBOT TEAM EFFORT: PROBLEM AND RECHARGE FOCUS

In Fig. 17, information on the interactions that occur in the simulation between the worker and team leader cognitive robot agents is shown. While moving towards its next rock sample task, a worker robot’s remaining battery charge drops below the predefined threshold; the worker robot then sends a *problem report* to the team leader robot. The content of the report indicates a low battery. The team leader robot immediately sends a reply confirming receipt of the worker robot’s problem message. The team leader robot then instructs the worker robot to set up the *recharge procedure*. Once the worker robot has acknowledged that it is ready to begin a recharge procedure, the team leader

Timestamp	Sender ID	Receiver ID	Protocol Name	Phase	Performative	Context
Fri Feb 27 13:37:39 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	problem_report	problem_in_mission	inform	problem
Fri Feb 27 13:37:40 EST 2009	simagent.hasioneteamlead000	simagent.hasioneworker000	problem_report	problem_in_mission	ack	problem

Timestamp	Sender ID	Receiver ID	Protocol Name	Phase	Performative	Context
Fri Feb 27 13:37:40 EST 2009	simagent.hasioneteamlead000	simagent.hasioneworker000	recharge_protocol	recharge_setup	inform	recharge
Fri Feb 27 13:45:36 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	recharge_protocol	recharge_setup	ack	recharge

Timestamp	Sender ID	Receiver ID	Protocol Name	Phase	Performative	Context
Fri Feb 27 13:27:41 EST 2009	simagent.hasioneteamlead000	simagent.hasioneworker000	team_work_effort	initiate_mission	query	mission_load
Fri Feb 27 13:27:41 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	team_work_effort	initiate_mission	reply	mission_load
Fri Feb 27 13:27:42 EST 2009	simagent.hasioneteamlead000	simagent.hasioneworker000	team_work_effort	initiate_mission	directive	mission_load
Fri Feb 27 13:27:42 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	team_work_effort	initiate_mission	ack	mission_load
Fri Feb 27 13:28:35 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	team_work_effort	initiate_mission	inform_done	mission_load
Fri Feb 27 13:28:35 EST 2009	simagent.hasioneteamlead000	simagent.hasioneworker000	team_work_effort	execute_mission	directive	start_task
Fri Feb 27 13:28:36 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	team_work_effort	execute_mission	ack	start_task
Fri Feb 27 13:45:37 EST 2009	simagent.hasioneteamlead000	simagent.hasioneworker000	team_work_effort	interrupt_mission	directive	interrupt
Fri Feb 27 13:45:37 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	team_work_effort	interrupt_mission	ack	interrupt
Fri Feb 27 13:45:37 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	team_work_effort	interrupt_mission	inform_done	interrupt

FIG. 17 Screenshot of worker agent interacting with team leader agent as a low battery problem occurs. Note the multiple conversations and protocol efforts that are involved.

Timestamp	Sender ID	Receiver ID	Protocol Name	Phase	Performative	Context
Fri Feb 27 13:37:40 EST 2009	simagent.hasioneteamlead000	simagent.hasioneworker000	recharge_protocol	recharge_setup	inform	recharge
Fri Feb 27 13:45:36 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	recharge_protocol	recharge_setup	ack	recharge
Fri Feb 27 13:45:37 EST 2009	simagent.hasioneteamlead000	simagent.hasioneworker000	recharge_protocol	recharge_setup	directive	proceed_to_location
Fri Feb 27 13:45:37 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	recharge_protocol	recharge_setup	ack	proceed_to_location
Fri Feb 27 14:24:23 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	recharge_protocol	recharging	inform	charging_station
Fri Feb 27 14:24:24 EST 2009	simagent.hasioneteamlead000	simagent.hasioneworker000	recharge_protocol	recharging	ack	charging_station
Fri Feb 27 14:35:06 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	recharge_protocol	recharging	inform_done	charging_station
Fri Feb 27 14:35:06 EST 2009	simagent.hasioneteamlead000	simagent.hasioneworker000	recharge_protocol	post_recharge	directive	proceed_to_location
Fri Feb 27 14:35:06 EST 2009	simagent.hasioneworker000	simagent.hasioneteamlead000	recharge_protocol	post_recharge	ack	proceed_to_location

FIG. 18 The worker is fully involved in the recharge procedure, although the protocol conversation regarding the teamwork effort is on interrupted status. Note the different phases of the recharge protocol.

robot orders the worker robot to *interrupt* its active mission. In these protocol-driven conversation exchanges, three protocols have been utilized, some concurrently, in three different conversational exchanges: the problem report protocol, the teamwork effort protocol and the recharging protocol. Note that the teamwork activity for this robot has been successfully interrupted, but not canceled. Not shown is the use of a fourth protocol; the worker robot *cancels* its claim to the rock sample area via the proximity avoid protocol.

The simulation unfolds in Fig. 18 as the team leader orders the worker to proceed to the recharging station. The worker robot acknowledges the order and proceeds to the station. The worker notifies the leader that it has arrived at the charging station. The team leader acknowledges the worker’s arrival. The worker docks with the recharging station and begins recharging. When the recharging process is completed, the worker robot has the initiative in this phase of the protocol; it notifies the team leader robot that is has completed the recharging. To start the next phase (postrecharge) of the recharging protocol, the team leader then orders the worker robot to return to its rock sampling area; this is acknowledged by the worker. The simulation continues to unfold, ending in the worker robot back in its work area and waiting. It informs the team leader that it is finished with the recharging process; the team leader then directs it to *resume* the mission. Figure 19 provides screen captures of the worker robot in its task area and as it moves towards the recharger, as just described and shown in Figs. 17 and 18.

Note that this simulation is one thread of activity that could occur. The protocols are flexible; the team leader can make different choices as to the resolution of the problem. For example, it could order the worker robot to interrupt its mission and proceed to a location at the rock cart. The team leader could then terminate the mission for that worker robot; this might occur if the leader had received

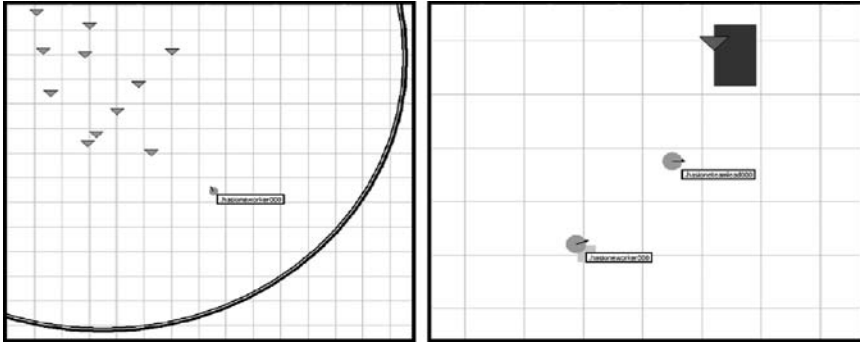


FIG. 19 In the left screen capture, the worker robot is active in its rock sampling task area; in the right screen capture the worker robot has moving to the recharge station.

word from an astronaut or a softbot that the mission would end early. Also note that the protocols may be utilized in an interleaved manner. Protocol-specific conversations organize the interactions; the decision logic determines which branch of the protocol to advance in what manner at a given time. The decision logic can take into account multiple areas of concern, including environmental information or external (to the team) messages. The decision logic is extensible.

VII. PROTOCOL UTILIZATION BY THE DEPLOYED ROBOT TEAM

A. LABORATORY ENVIRONMENT AND THE PHYSICAL ROBOT

The physical environment for robot team deployment is the laboratory environment. The teamwork and safety elements of the notional scientific sampling mission, as supported by protocols, were investigated in hardware in a laboratory setting.

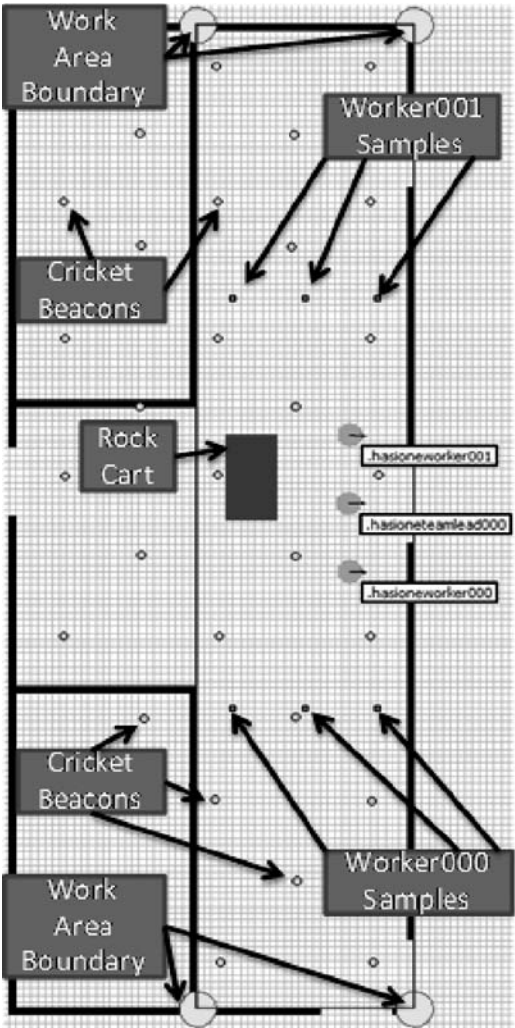
A map of the laboratory, shown in Fig. 20, replaces the notional lunar map of Fig. 9. In addition to the physical layout of the space, a key feature is the Cricket localization system (data available online at <http://www.cricket.csail.mit.edu>). These are indicated in circles on the map; their three-dimensional locations are known within the laboratory framework. The Cricket system allows the robots to know their position within the laboratory frame. Kulis et al. [27] describe the use of both the Cricket range measurements and odometry data (from robot wheel revolutions) via a Kalman filter formulation to generate position data. The laboratory has a wireless network for experimentation; robot communications (which are protocol driven) are sent and received via this network.

The rocks are Styrofoam cubes that are covered in either green or pink tape. The green rocks are not of scientific interest whereas the pink rocks are those that should be collected and brought by a worker robot to the rock cart. The rocks are

termed “worker samples” in Fig. 20. In the laboratory space, the rock cart is actually a rectangular area on the floor. It is shown in Fig. 21.

Figure 21 shows a schematic of the Amigo-bot hardware platform (the environment) for the cognitive robot agents. The laptop hosts the protocol-aware robot agent, which executes in the context of the DCF and the Cybele™ agent platform, thus providing the “brains” of the Amigo-bot. Note the Cricket beacons located on the Amigo-bot. A passive C-shaped collector is affixed to the lower front of the Amigo-bot’s frame. The Amigo-bot will use this collector to capture a lunar sample block that must be returned to the laboratory’s rock cart. Also note the

camera that is mounted on the Amigo-bot’s frame. The camera is utilized in identifying a block; pink blocks are of interest and are returned to the cart whereas the green blocks are not of interest and are not collected.



B. INCORPORATION OF PHYSICAL ENVIRONMENT ELEMENTS INTO THE SIMULATION

Before moving the protocol library and the cognitively enhanced robot agent code, developed through software simulation, to hardware, a further set of simulations that take place in the laboratory environment was performed. In addition to the laboratory environment, details of hardware and sensor aspects of the robot are incorporated into the simulations. For example, the driver code to capture the camera images is incorporated into the

FIG. 20 Diagram of the laboratory environment.

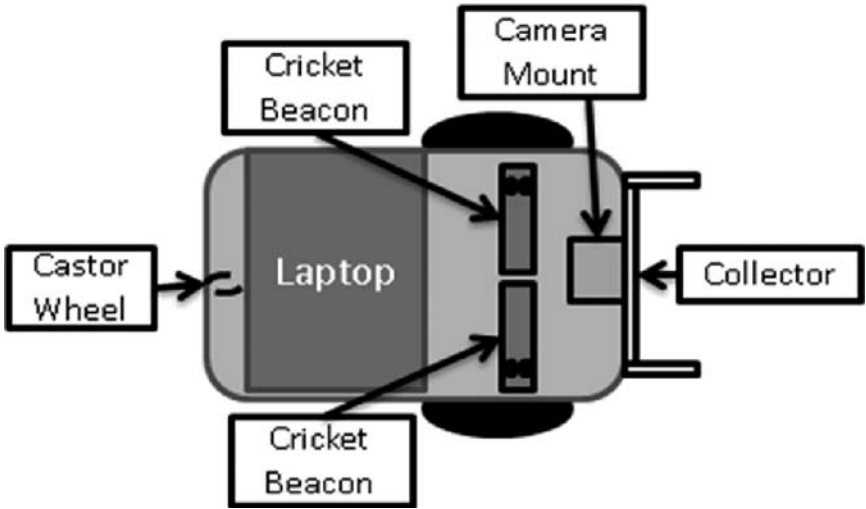


FIG. 21 Diagram of the Amigo-bot hardware platform for the cognitive robot agent.

simulation and the state estimator activity in the cognitive robot agent uses the Kalman filter plug-in [27].

However, additional changes were made to the cognitively enhanced robot agent entity that is being simulated in order to ready it for deployment to hardware; these changes are 1) utilization of images captured by the camera and 2) redesign of the lower-level robot controls that will enable it to acquire and move a block with the C-shaped plastic collector that functions as a plow.

To support the first change, many images were taken of the block, from a camera that was located at the height it would have when mounted on the Amigo-bot frame (see Fig. 22). A set of images was produced at different (x, y) distances of the camera from the block to form an image library. In the simulation for deployment, an appropriately distanced image of a block is provided to the robot, when it is “facing” the block.

The second change is necessary because the cognitive robot agent no longer has a notional



FIG. 22 Processed image of blocks: although shown in grey, one block is colored bright green and the other bright pink.

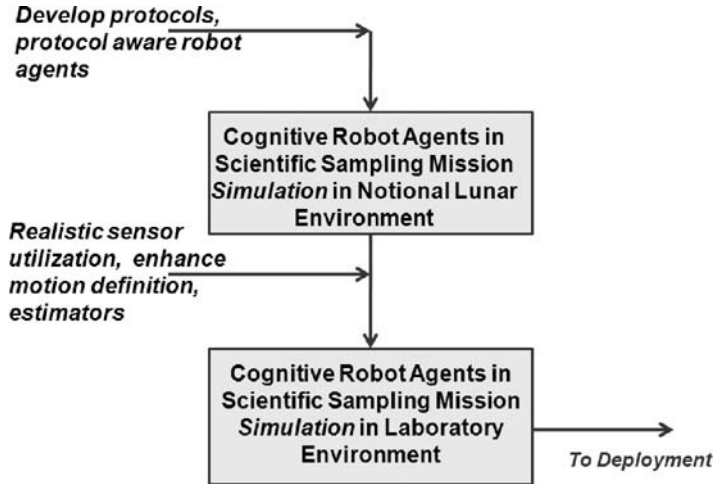


FIG. 23 Process to develop deployable robotic software for hardware experiments in the laboratory environment.

gripper in the lunar environment; rather, it has a plastic collector that keeps a block captured as the robot “plows it” to the rock cart area. The lower-level motion behaviors are a “deployed” version of those used for the notional lunar environment. This involves additional constraints on the motion. For example, in the Deployed Collect and Analyze Sample, there are camera positioning and a plow positioning constraints. A worker robot must have a view of the sample from its onboard camera in order to be able to identify the sample. If the worker robot determines the rock is pink and must be collected, then it must collide with the sample, keeping the sample between the boundaries of the C-shaped plow.

The simulations for deployment were performed in preparation for the laboratory experiments to test whether the protocols were workable on hardware systems. Figure 23 illustrates the overall process leading to deployment of a protocol-aware autonomous robot team: 1) development of the protocols, 2) simulations in the notional lunar environment to test the protocols, 3) transition of simulation to those for the laboratory frame, with modifications due to realistic sensor utilization and improved motion estimation, and 4) deployment.

C. DEPLOYMENT RESULTS: AUTONOMOUS ROBOTS ENGAGED IN THE SCIENTIFIC SAMPLING MISSION

Figures 24–29 present video frames of the cognitive team leader robot agent coordinating the activities of two cognitive worker robot agents as they perform the



FIG. 24 The leader orders both workers to begin their mission by proceeding to their respective mission start locations.

rock sampling mission in the laboratory environment. Only pink rocks (blocks) are to be collected; green ones are rejected.

The robots communicate through a wireless network that was set up in the laboratory for this experiment. All robots communicate via this network. All messages were logged, with timestamps, message header, and content



FIG. 25 Worker 000 arrives at its mission start location first and notifies the team leader.

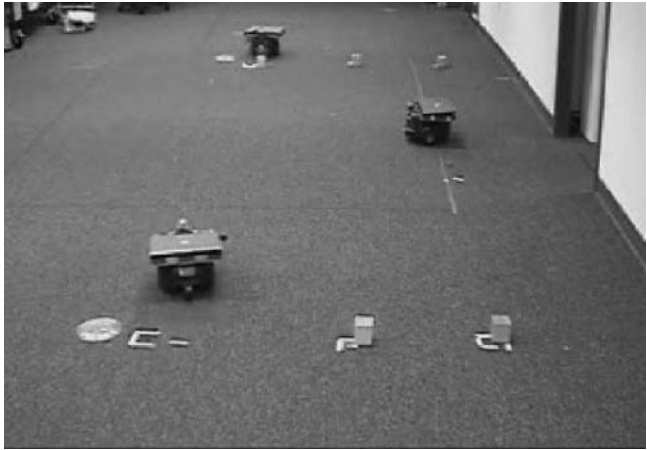


FIG. 26 Worker 001 finishes processing its first rock before Worker 000.

information. The message header information included all protocol details as well as the sender ID and timestamps. After the experiments, the detailed information was taken from robot's laptop.

In Fig. 24, Worker 001 is the top-most robot, the team leader is located in the middle, and Worker 000 is located closest to the bottom of the screen.

In Fig. 25, after being notified of the worker's location, the team leader immediately instructs the worker to begin its mission by starting the first task.

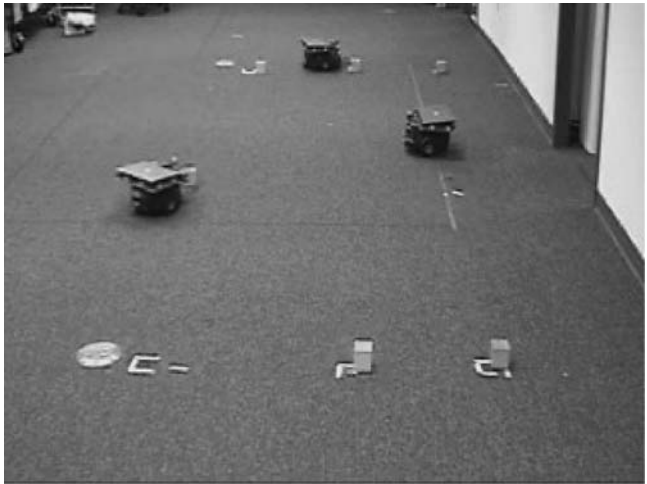


FIG. 27 Worker 000 finishes processing its first rock and takes it to the rock cart area.

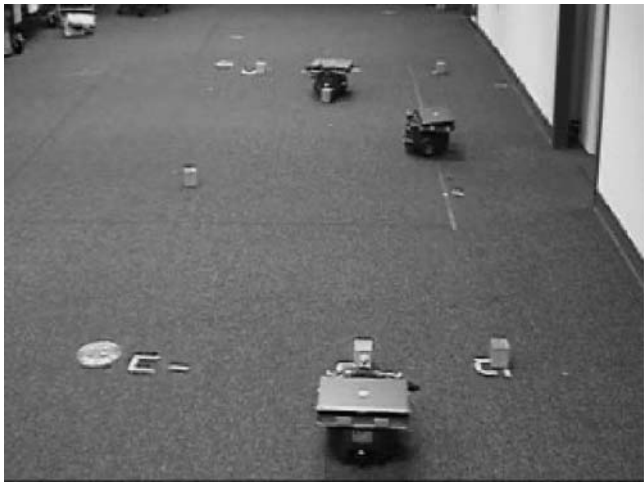


FIG. 28 Worker 000 finishes processing its second task; it has rejected the block of the "wrong" color.

Once Worker 000 has received the directive to start its mission, the Worker 000 notifies all other entities to avoid its work area.

In Fig. 26, Worker 001 sends a status update to the team leader, which the team leader immediately acknowledges.

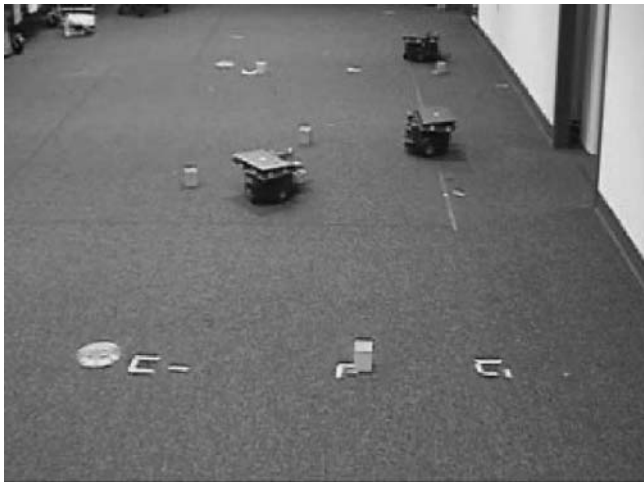


FIG. 29 Worker 000 completes its final task and notifies the team leader. The team leader acknowledges the status update, and that the worker's mission is completed.

In Fig. 27, Worker 000 notifies the leader that it has completed the task. The leader immediately acknowledges the message.

In Fig. 28, Worker 000 finishes processing its first rock. This is indicated by the fact it has taken the rock (block) to the rock cart.

The rock cart area is denoted by the rectangular area that is near the team leader robot. Worker 000 notifies the team leader that it has completed the task. The team leader immediately acknowledges the message.

In Fig. 28, Worker 000 finishes processing its second task; it has rejected the block of the “wrong” color. The Worker 000 notifies the team leader of the task completion. The team leader immediately acknowledges the receipt of the status update.

In Fig. 29, Worker 000 has completed its final task and notifies the team leader of this. The team leader acknowledges the status update, and that the worker’s mission is completed.

The team leader orders the worker to terminate its mission. Worker 000 cancels its work area proximity avoid and confirms to the leader that it has terminated its mission.

The mission has not been terminated yet for the robot Worker 001. It has not finished its sample assay of the rocks.

Note that the proximity avoid protocol was modified for the deployment experiments. As the laboratory distance was relatively short, there was not a large enough travel area between the cart and the rock area to warrant cancellation of the claim. In this case, each worker robot’s will cancel its claim only after its mission is complete.

D. DEPLOYMENT RESULTS: ASTRONAUT WALK THROUGH NEAR AUTONOMOUS ROBOTS ENGAGED IN THE SCIENTIFIC SAMPLING MISSION

Figures 30–34 show a worker robot safing itself using the safing protocol when the astronaut walks through the mission task area in the laboratory environment. The worker robot stops its efforts until the astronaut is out of the area.

Note that the astronaut is pushing a cart in which an Amigo-bot is carried. The reason for this is that the astronaut must participate in the communication link with the workers. Thus, the Amigo-bot robot is sending out the astronaut’s ID.

Figure 30 shows the team leader robot in the vicinity of the rock cart. In this frame, the worker robot has just captured the pink block for collection and placement in the rock cart. Figure 31 shows that the Worker robot has started to move to the rock cart. The astronaut is just coming into the mission task area.

Figure 32 shows that as the astronaut comes further into the mission task area, the worker robot has stopped its motion toward the rock cart. It has safed itself.

Figure 33 shows that as the astronaut walks through the mission task area, the worker robot remains motionless.



FIG. 30 The worker has just collected a rock sample to return to the rock cart area.

In Fig. 34, the astronaut has passed well out of the mission task area. The worker robot has had time to desafe itself and to return to its task activity. It has deposited its rock in the cart (lined area) and is backing away from the rock before returning to continue its task activity of rock sampling.



FIG. 31 The worker is approaching the boundary of the Rock Cart area.



FIG. 32 The astronaut walks into the mission task area.

Note that the backing away from a rock rather than turning immediately is a low-level behavior that is incorporated via “motion atoms” using the MDLe. This motion was developed for the deployed physical robots because each Amigo-bot has a C-shaped collector that is not retractable. The low-level motion allows the worker robots to avoid hitting the blocks without intent.



FIG. 33 The astronaut walks through the mission area.



FIG. 34 After the astronaut has left the area, the worker robot desafes itself and continues its mission.

VIII. DISCUSSION AND CONCLUSIONS

This effort has utilized the abstraction of a Mission Society in elucidating behavior norms to inform protocol development for interactions involving autonomous entities and systems, including astronauts, multi-agent systems, and robot teams. The protocols support mission activity and safety.

Values of the Mission Society include the critical value placed on human crew safety and the high value placed on mission completion and success. Behavior norms that support the values were discussed; these involve ramifications regarding how autonomous entities such as multi-agent systems and robot teams could participate in the Mission Society. Among the ramifications are 1) the need for a multi-agent system to provide access to information on its subsystem functioning and any changes; 2) the need for an autonomous entity team to utilize relevant information to guide its activity selection; 3) the need for a robot team to provide information on its task status; 4) the need for a representative of the autonomous entity to interact with the astronaut crew, leading to role-based capability within the autonomous entity teams; 5) the ability of a multi-agent system or robot team to relinquish its autonomous activity at the command of an astronaut; 6) the need for safety in autonomous team functioning, especially when changing autonomy levels or, in the case of a robot team, when physically near to a human; and 7) teamwork orientation, with roles and potential command structure within an autonomous system.

Two different exploration mission scenarios were utilized to ground the investigation. The first focused on a long-duration mission in which multiple multi-agent systems representing vehicle subsystems or mission functions interacted with the astronaut and, as appropriate, with each other. The second focused on a lunar scientific sampling mission and involved an autonomous robot team.

Investigation of the long-duration mission scenario resulted in the development of a set of requirements on each participating multi-agent system. Each MAS may have different capabilities in different autonomy modes. The capabilities and their mapping to autonomy levels are determined at MAS design. Requirements on MAS design include the need to expose descriptive information as to its autonomy level, performance and safety status, the ability to move to different autonomy modes, and the ability to interact with an astronaut in the appropriate autonomy mode consistent with MAS functionality. The roles needed to be instantiated within each MAS: the coordinator, the descriptor, and the designee. The coordinator role is responsible for managing the MAS agents in situations involving autonomy changes. The designee role must interact with the astronaut in the no-autonomy mode (if functionality in that mode is offered by the MAS). The descriptor role must answer queries concerning any descriptive information that each MAS must expose. Note that the details of the descriptive information will change with circumstances.

A primary protocol developed for the long-duration mission scenario is an astronaut-ordered autonomy mode change. Additional protocols, including the description protocol, an LRU replacement dialogue protocol that serves as a template for other dialogue intensive interactions, and startup protocols for virtual teams and for each participating MAS, were developed.

MAS information exposure and protocol support for an astronaut to interact with multiple MAS were prototyped as a two-console system. This includes MAS dependency visualization, hidden protocol support to allow the astronaut to easily send a query or autonomy change message to (an agent in) a MAS as well as dialogue support.

Protocol development for an autonomous robot team was motivated by the scientific sampling mission. Within the team, two roles were utilized, that of team leader and worker. The team leader role was the designated role to interact with the astronaut when the astronaut was engaged in oversight of the scientific sampling efforts. Within the scientific sampling mission scenario, the astronaut could choose to view the robots as individuals rather than as team members. To support this, protocols were developed to allow the astronaut to elect to order a robot to safe or desafe and to query its status. However, the primary scenario focus was that of an autonomous team, potentially under oversight.

Protocol support for the sampling mission was centered on the teamwork effort protocol (TWEP). However, ancillary protocols must be utilized in concert with the TWEP; these allow for robot coordination when nontask concerns arise within the environment. These can include a hardware problem, a battery recharge need, etc. The robots, in acting autonomously, must be safety

conscious; this has also been supported by protocol development. Both problem and safety protocols were developed. Detailed of protocols have been presented. Protocols were prototyped in simulations and then utilized in a deployed hardware-based cognitively enhanced robot agent team.

In this effort, the cognitively enhanced, protocol-aware robot agent was developed. Each cognitively enhanced robot agent has a sense of agency [22]; it encompasses the characteristics of autonomy, task orientation, communicativeness, and cooperativeness (i.e., social). The communicativeness is supported via the protocol set. Through the use of the *protocol set* and the *decision structures*, the autonomous robot team is adaptive to external conditions or problems that may occur.

In the development of a protocol-aware, cognitively enhanced robot agent, a finite state machine representation (FSM) for the decision structures within the decision manager was employed. This representation is also utilized in formulations of lower-level motion behaviors (MDLe). Reference [31] notes the FSM representation provides a suitable coding pattern for agent and robot behaviors and utilizes it. The protocol structures developed in this were also formulated as a set of finite state machines. This common representation facilitated a natural synergy between the protocol set and the cognitive decision support architecture. Results of decisions form a portion of guard conditions in the protocol FSM. Note that a protocol-focused module structure was the design chosen for the decision manager software architecture.

Such protocol-aware, cognitively enhanced robot agents formed the autonomous team. These were able to maintain multiple “conversations,” related to not only direct mission activity but also to the ancillary elements that have a strong impact on mission success. These include checks to make sure all robots on the team have loaded the proper mission information, handling issues related to equipment status/malfunctions or the need for recharging batteries, the possibility of mission interruption due to unforeseen external events, etc. An ability to share such information in an unambiguous manner provides the autonomous robot team with flexibility in meeting mission objectives. Currently such flexibility is provided by humans who exercise control of robots.

Hirsch and colleagues [8] have investigated the concept of an extravehicular robotic assistant (ERA). Part of their effort focused on how a robotic assistant's communications with an astronaut should be structured. The robot control architecture used in the ERA project is oriented towards the astronaut in command. The human tells the robot to “perform a certain task and the robot works on that task until finished or in receipt of another command from the human who interrupts that task.” This is in contrast to the current work which developed and investigated a protocol-driven approach in order to support coordinated autonomous team interactions among the robot agent members of a robotic team. In contrast with the approach of [8], the team leader robot agent has the flexibility of modifying the tasks of worker robot on the team. Although the notion of agency and communication between an astronaut and a robot is integral

to the work of [8], it does not have a focus on autonomous robot agent *team* coordination.

Bradshaw and colleagues [32] take a coordination-centric view. They focus on situations involving human, soft-bot, and robot agent team activity. They utilize the concept of a coordination policy. These are used explicitly, for example, "an agent/robot must notify others when its task is finished." This is similar to our use of mission behavior norms in developing the protocol set. We note that our extension to an astronaut oversight of the teamwork effort protocol utilized conversational coordination points to support the interactions of the team leader robot and the astronaut; this is an additional exemplar involving the use of coordination points, as in [32].

Studies of the protocols in a physical robot agent team deployed in the laboratory were performed. Two worker robots and a team leader robot completed the scientific sampling mission. In an additional scenario thread involving an astronaut walk-through of the task area, the worker robot safed itself and did not resume its task activity until the astronaut had left the area.

Although the protocols that have been developed are suitable for general use in mission-oriented robotic task activities, they have been influenced by behavior norms for space exploration missions.

ACKNOWLEDGMENT

Support from NASA SBIR Contract NNJ08JA64C is gratefully acknowledged.

REFERENCES

- [1] Schreckenghost, D., Thronesbery, C., Bonasso, P., Kortenkamp, D., and Martin, C., "Intelligent Control of Life Support for Space Missions," *IEEE Intelligent Systems*, Vol. 17, No. 5, 2002, pp. 24–31.
- [2] Lyell, M., Krueger, W., Zhang, G., Alena, R., Xu, R., and Haynes, L., "Agent-Based Approach to Health Monitoring Systems Applied to ISS Power System," AIAA Paper 2007-0761, Jan. 2007.
- [3] Allen, C. C., Lofgren, G. E., Treiman, A. H., and Lindstrom, M. L., "2007 Sample Curation at a Lunar Outpost," NASA Advisory Council Workshop on Science Associated with Lunar Exploration Architecture, Feb.-March 2007, <http://www.lpi.usra.edu/meetings/LEA/whitepapers/index.shtml>.
- [4] Shearer, C., "Importance of Sample Science-Sample Return and Potential Sample Targets 2007," NASA Advisory Council Workshop on Science Associated with the Lunar Exploration Architecture, Feb.-March 2007, <http://www.lpi.usra.edu/meetings/LEA/whitepapers/index.shtml>.
- [5] Clancey, W. J., Sachs, P., Sierhuis, M., and van Hoof, R., "Brahms: Simulating Practice for Work Systems Design," *International Journal of Human-Computer Interaction*, Vol. 49, No. 6, 1998, pp. 831–865.

- [6] Rabideau, G., Knight, R., Chien, S., Fukunaga, A., and Govindjee, A., "Iterative Repair Planning for Spacecraft Operations in the ASPEN System," *International Symposium on AI, Robotics and Automation for Space*, 1999, pp. 99–107, <http://robotics.estec.esa.int/i-SAIRAS/isairas1999/mainmenu.pdf>.
- [7] Fong, T., Kunz, C., Hiatt, L., and Bugajska, M., "The Human-Robot Interaction Operating System," *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, edited by A. C. Schultz and D. J. Bruemmer, ACM Press, New York, 2006, pp. 41–48.
- [8] Hirsh, R., Graham, J., Tyree, K., Sierhuis, M., and Clancey, W., "Intelligence for Human-Assistant Planetary Surface Robots," *Intelligence for Space Robotics*, edited by A. M. Howard and E. W. Tunstel, TSI Press, Albuquerque, NM, 2006, pp. 261–279.
- [9] Bradshaw, J., Acquisti, A., Allan, J., Breedy, M., Bunch, L., Chambers, N., Galescu, L., Goodrich, M., Jeffers, R., Johnson, M., Jung, H., Kulkarni, S., Lott, J., Olsen, D., Sierhuis, M., Suri, N., Taysom, W., Tonti, G., Uszok, A., and Van Hoof, R., "Team-Work Centered Autonomy for Extended Human-Agent Interaction in Space Applications," *Proceedings of the AAAI 2004 Spring Symposium*, AAAI Press, Stanford, 2004, pp. 22–24.
- [10] Schreckenghost, D., Martin, C., Bonasso, P., Kortenkamp, D., Milam, T., and Thronesbery, C., "Supporting Group Interaction Among Humans and Autonomous Agents," *Connection Science*, Vol. 4, No. 4, 2002, pp. 361–369.
- [11] Fong, T., Bualat, M., Edwards, L., Fluckinger, L., Kunz, C., Lee, S., Park, E., To, V., Utz, H., Ackner, N., Armstrong-Crews, N., and Gannon, J., "Human-Robot Site Survey and Sampling for Space Exploration," AIAA Paper 2006-7425, Sept. 2006.
- [12] Fong, T., and Nourbakhsh, I., "Interactions Challenges in Human-Robot Space Exploration," *ACM Interactions*, 2005, pp. 42–45.
- [13] Dorais, G., and Kortenkamp, D., "Designing Human-Centered Autonomous Agents," *Lecture Notes in Computer Science*, Vol. 2112, Springer-Verlag, New York, 2000, pp. 321–324.
- [14] Lyell, M., and Drozd, W., "Human Autonomous System Interaction Framework to Support Safety in Astronaut-Robot Team Interactions," AIAA Paper 2010-764, Jan. 2010.
- [15] Lyell, M., and Drozd, W., "Protocol-Aware Enhanced Cognition Robot Agent Design for Team Work Effort in Lunar Exploration Missions," 2010 Workshop on Practical Cognitive Agents and Robots, AAMAS PCAR 2010, ACM Special Interest Group on Artificial Intelligence and International Foundation for Autonomous Agents and Multiagent Systems, May 2010.
- [16] Lyell, M., Grinberg Webb, A., Nanda, J., and Chen, W., "Human-Autonomous System Interaction Framework to Support Astronaut-Multi-Agent System Interactions," AIAA Paper 2009-428, Jan. 2009.
- [17] Sandal, G., "Psycho Social Issues in Space: Future Challenges," *Gravitational and Space Biology Bulletin*, Vol. 14, No. 2, 2001, pp. 47–54.
- [18] Hexmoor, H., and Vaughn, J., "Computational Adjustable Autonomy for NASA Personal Satellite Assistants," *ACM Proceedings of the 2002 ACM Symposium on Applied Computing*, edited by H. Haddad, G. Papadopoulos, and B. Panda, ACM Press, New York, 2002, pp. 21–26.
- [19] Searle, J., *Speech Acts an Essay in the Philosophy of Language*, Cambridge Univ. Press, Cambridge, England, U.K., 1969.

- [20] "FIPA ACL Message Structure Specification," Foundation for Intelligent Physical Agents, Doc. No. SC00061G, Geneva, http://www.fipa.org/specs/fipa00061/SC00061G.html#_Toc26669700 [retrieved 26 Oct. 2009].
- [21] Kremer, R., and Flores, R., "Flexible Conversations Using Social Commitments and a Performatives Hierarchy," *Lecture Notes in Artificial Intelligence*, Vol. 3589, Springer-Verlag, New York, 2006, pp. 93–108.
- [22] Wooldridge, M., and Jennings, N. R., "Intelligent Agents: Theory and Practice," *The Knowledge Engineering Review*, Vol. 10, No. 2, 1995, pp. 115–152.
- [23] Jennings, N. R., "An Agent-Based Approach for Building Complex Software Systems," *Communication of the ACM*, Vol. 44, No. 40, 2001, pp. 35–41.
- [24] Zambonelli, F., Jennings, N., and Wooldridge, M., "Developing Multi-Agent Systems: the Gaia Methodology," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 12, No. 3, 2003, pp. 317–370.
- [25] Griss, M., and Pour, G., "Accelerating Development with Agent Components," *Computer*, Vol. 34, No. 5, 2001, pp. 37–43.
- [26] Lyell, M., "Final Report: An Integrated Human Systems Interaction (HSI) Framework for Human-Agent Team Collaboration," NASA SBIR Contract NNJ07JB29C, July 2007.
- [27] Kulis, Z., Manikonda, V., Azimi-Sadjadi, B., and Ranjan, P., "The Distributed Control Framework: A Software Infrastructure for Agent-Based Distributed Control and Robotics," *American Control Conference*, IEEE, Piscataway, NJ, 2008, pp. 1329–1336.
- [28] Lyell, M., Levy, R., and Satapathy, G., "Different Types of Software Agents in the Cybele Software Agent Framework," *Agent Link*, Vol. 18, No. 8, 2005, pp. 40, 41.
- [29] Erol, K., Lang, J., and Levy, R., "Designing Agents from Reusable Components," *Proceedings of the 4th International Conference on Autonomous Agents*, edited by C. Sierra, M. Gini, and J. S. Rosenschein, ACM Press, New York, 2000, pp. 77, 78.
- [30] Manikonda, V., Krishnaprasad, P. S., and Hendler, J., "Languages, Behaviors, Hybrid Architectures and Motion Control," *Mathematical Control Theory*, edited by J. Baillieul and J. C. Willems, Springer-Verlag, New York, 1998, pp. 199–226.
- [31] Novak, P., and Jamoroga, W., "Code Patterns for Agent-Oriented Programming," *Proceedings of the AAMAS 2009*, edited by K. Decker, J. Sichman, C. Sierra, and C. Castelfranchi, International Foundation for Autonomous Agents and Multiagent Systems, SC, 2009, pp. 105–112.
- [32] Bradshaw, J., Feltovich, P., Johnson, M., Breedy, M., Bunch, L., Eskridge, T., Hyuckchul, J., Lott, J., Uszok, A., and Diggelen, J., "From Tools to Teammates: Joint Activity in Human-Agent-Robot Teams," *Human Centered Design*, edited by M. Kurosu, LNCS 5619, Springer-Verlag, New York, 2009, pp. 935–944.

Framework for User–Guided Search and Adaptive Target Tracking via Cooperative UAVs

R. A. Cortez,^{*} D. Tolić,[†] and I. Palunko[‡]

University of New Mexico, Albuquerque, New Mexico 87131

N. Eskandari^{††}

University of British Columbia, Vancouver, British Columbia V6T 1Z4, Canada

M. Oishi,[§] R. Fierro,[¶] and I. Wood^{}**

University of New Mexico, Albuquerque, New Mexico 87131

I. INTRODUCTION

One of the main advantages of using unmanned aerial vehicles (UAVs) is their ability to adaptively reconfigure the system based on information from the environment being monitored. The maneuverability of UAVs make them ideal platforms for surveillance monitoring and target tracking, especially in challenging environments such as found in urban settings, which may have a cluttered or dynamic search space. The goal of this chapter is to provide a collaborative framework for efficient deployment of mobile cooperative sensing networks in autonomous or semi-autonomous operation. In this chapter we integrate a high-level cooperative path planning strategy for target search and tracking with a low-level real-time controller that guides the team of UAVs, in order to develop a novel framework for efficient and robust information processing, and decision-making in cooperative networks.

The topic of networked UAVs for target search and tracking, although not entirely new, has begun to gain interest in the controls community. Several

^{*}Graduate Student, Department of Mechanical Engineering.

[†]Graduate Student, Department of Electrical and Computer Engineering.

[‡]Graduate Student, Department of Electrical and Computer Engineering.

[§]Assistant Professor, Department of Electrical and Computer Engineering.

[¶]Associate Professor, Department of Electrical and Computer Engineering.

^{**}Professor, Department of Mechanical Engineering.

^{††}Graduate Student, Department of Mechanical Engineering.

approaches to reconfiguring the UAV network have been investigated. A technique for dynamically reconfiguring search spaces in order to enable Bayesian autonomous search and tracking missions with moving targets is presented in [1]. The authors in [2] develop a framework for a group of fixed-wing UAVs to cooperatively search and localize static targets using Bayesian filtering techniques and receding-horizon path planning. A somewhat similar work to [2] can be found in [3]. The difference is that [3] takes into account a dynamical model of quadrotors (i.e., UAVs), investigates several sensing modalities, and puts more emphasis on developing scalable distributed estimation techniques. Particle filtering under intermittent information for multi-agent scenarios is investigated in [4]. The aforementioned approaches use probabilistic methods that enable planning techniques based on minimization of entropy (i.e., uncertainty). Drawbacks of the Bayesian inference can be found, for example, in [5] or in an interesting discussion initiated in [6].

The work in [7] presents a rule-based intelligent guidance strategy for autonomous pursuit of mobile targets using UAVs in an area with threats, obstacles, and restricted regions. A least-square estimation and kinematic relations are used to estimate/predict the target states based on noisy position measurements. However, only generic and simplified models/behaviors of both targets and UAVs are considered. A tightly integrated systems architecture for a decentralized cooperative search, acquisition, and track (CSAT) mission management algorithm along with actual hardware flight tests is presented in [8]. In a complimentary paper [9], the authors propose an architecture similar to the one we propose in this chapter, but do not consider the shared control scenario and assume simplified target models.

The novelty of our proposed framework is twofold: 1) integration of path planning/UAV coordination with real-time control and 2) incorporation of human-in-the-loop operation via high-level control inputs. The latter capability is particularly important in search and surveillance scenarios, where human intuition may improve the efficacy or speed of target acquisition. Although most existing work considers high-level coordination and low-level control to be independent problems, our approach combines both into a single framework. We believe that such a framework has the potential to significantly improve overall reliability and enable new functionalities in UAV systems through design that is known to be correct *a priori*. In addition, our approach exploits both sophisticated and detailed UAV and target models as well as more general, high-level models that enable fast calculations in real time.

The chapter is organized as follows. After describing the problem formulation in Sec. II, Sec. III presents the path planning algorithms used to complete either (possibly human-guided) prioritized search or target tracking. Section IV describes the equations of motion of a quadrotor UAV as well as a low-level feedback linearizing control law. Section V presents our results in simulation and in hardware, and Sec. VI provides conclusions and directions for future work.

II. PROBLEM FORMULATION

A. PROBLEM DESCRIPTION

Consider a team of n kinematic agents (UAVs) as shown in Fig. 1, each equipped with a sensor/camera having a circular field of view (FOV). We denote the fleet of vehicles by $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$, and the FOV for vehicle j by \mathcal{B}_j . Each UAV is tasked with starting from some initial configuration and navigating to various regions within the area of interest (AOI), which contain the highest probability of containing a target of interest first, while avoiding collisions with other UAVs. We assume there exists a target of interest in the AOI. Once the target of interest is identified, the UAVs should track the identified target while minimizing their use of resources (sensor use and computational load). Because the target is not necessarily stationary, the vehicles may need to proceed with search and tracking more than once.

Let the probability-of-detection (POD) map reflect the probability of detecting a target over the AOI. Define $\beta > 0$ to be a parameter that reflects the reduction in the probability of detection map for points inside each robot's FOV and $\delta > 0$ that can be considered as a forgetting factor of past measurements. Consider the AOI, $Q \subset \mathbb{R}^3$, with boundary ∂Q , to be a simple convex polygon. Let us denote the POD map as $M(q, t)$, where $q \in Q$. Here we assume that Q and its boundary ∂Q are known a priori by all UAVs. From the high-level motion coordination perspective, each UAV, p_j is assumed to be holonomic with dynamics:

$$\dot{p}_j = u_j \quad j = 1, \dots, n \quad (1)$$

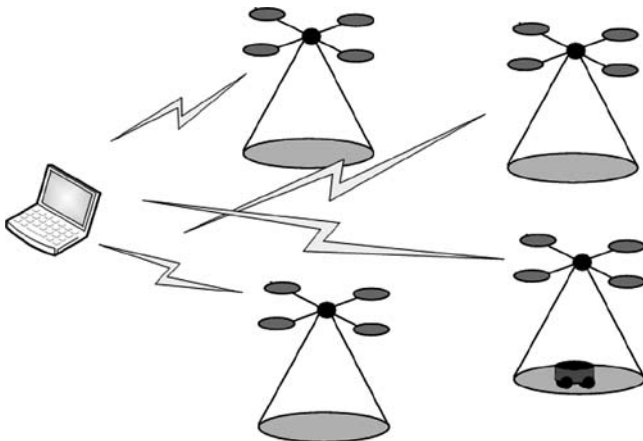


FIG. 1 Fleet of quadrotor UAVs within an AOI. Each vehicle j with state p_j has FOV \mathcal{B}_j and communicates to the fusion center.

where $u_j \in \mathbb{R}^3$ is the control input of agent j . From the low-level real-time controller perspective a 12-state dynamic model is used and presented in Sec. IV.A. In this formulation we assume that the UAVs are faster than the targets of interest and UAVs fly at a constant altitude above obstacles in the AOI. Also connectivity is assumed between the UAVs and the fusion center.

The target \mathcal{T} is assumed to be a unicycle unmanned ground vehicle (UGV) that performs purely stochastic maneuvers within the AOI; the dynamics are given by

$$\dot{x} = \begin{bmatrix} \dot{x}_{\mathcal{T}} \\ \dot{y}_{\mathcal{T}} \\ \dot{\theta}_{\mathcal{T}} \end{bmatrix} = \begin{bmatrix} v_s \cos \theta_s \\ v_s \sin \theta_s \\ \omega_s \end{bmatrix} \quad (2)$$

where ω_s and v_s are stochastic inputs given to the UGV such that $v_s \in [v_{\min}, v_{\max}]$ and $\omega_s \in [\omega_{\min}, \omega_{\max}]$. Modeling the target of interest as a unicycle with stochastic inputs allows us to address three main behaviors seen in pursuit evasion games: adversarial, nonadversarial, and collaborative.

The main problem is to design a hierarchical controller that allows the system, in collaboration with a user, to effectively track the dynamic target while conserving finite resources (e.g., battery power and computational time) and obeying state and input constraints.

B. PROPOSED APPROACH

Our approach is to develop a framework that combines the high-level motion planning of the group and the low-level control and stabilization of the UAVs. This approach enables the UAVs to bypass the coordination step and only have to communicate with a fusion center to obtain waypoint information. Although this may seem to be a drawback in the sense of a one-point failure, this approach allows for an easy transition from an autonomous mode to a semi-autonomous mode when human input may be needed. This is an important consideration in target

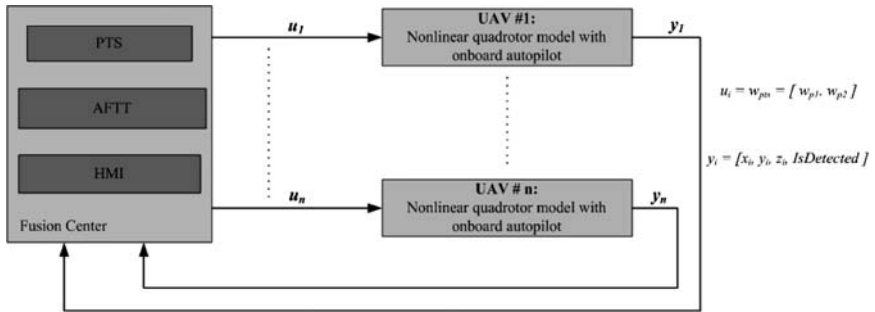


FIG. 2 Diagram of the framework the UAVs will employ to acquire and eventually capture a target of interest.

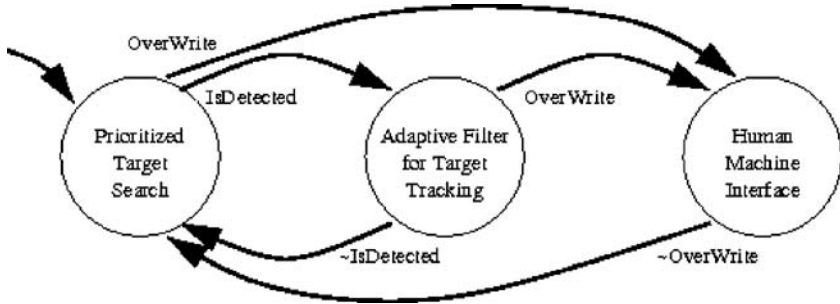


FIG. 3 Hybrid automata of hierarchical structure for collaborative prioritized search and target tracking. A false event flag is denoted by \sim .

search/tracking and surveillance scenarios where human intuition can play a vital role in mission completion. Figure 2 shows a diagram of how the framework is structured and implemented. We believe that allowing for a human-in-the-loop is an important element to our framework because of the benefit that human intuition and perspective can provide in search and surveillance scenarios.

To transition from one behavior to another, we introduce two boolean variables that act as flags for the fusion center. Initially the variable $IsDetected = \text{false}$ and the UAVs employ the prioritized target search (PTS) algorithm. When the target is detected by a UAV, $IsDetected = \text{true}$, and the fusion center switches controllers to the adaptive sampling-based filter for target tracking (AFTT).

The system initializes in the prioritized targeted search mode, a fully automated mode, but can switch into a human-guided mode, in which the human operator controls the UAV team manually through inputting desired waypoints. We denote this mode human-machine-interface (HMI) mode. At any point during the search or during the adaptive target tracking, the human can intervene, forcing the system into HMI mode.

The algorithm used by the fusion center to determine the appropriate mode of operation is shown in Fig. 3.

III. UAV COORDINATION

A. PRIORITIZED TARGET SEARCH

In [10], we propose a coordination algorithm that allows a team of sensor-enabled robots to navigate a region containing nonconvex obstacles and take measurements within the region that contain the highest probability of having “good” information first. This approach is motivated by scenarios where prior knowledge of the search space is known or when time constraints are present that limit the amount of area that can be searched by a robot team. Our cooperative control

algorithm combines Voronoi partitioning, a global optimization technique, and a modified navigation function to prioritize sensor detection. In this chapter we modify the prioritized sensing coordination algorithm to accommodate for sensing targets of interest rather than creating spatial distribution maps. We also modify the coordination algorithm to integrate the use of UAVs. With prior knowledge of the area and insight to where targets of interest may be, the coordination algorithm allows for prioritized searching of the AOI. Another difference that must be addressed is that of building spatial distribution maps vs target search. When building spatial distribution maps, sensing agents only need to visit a particular region once to take measurements because it is assumed that the environmental phenomenon being measured can be treated as static over the time period of the search. In the target search scenario, however, this is not the case because targets are moving throughout the AOI while UAVs are searching. Regions that have been searched before by UAVs may need to be searched again because of the dynamic nature of the target.

To address this problem, let the POD map $M(q, t)$ be updated in the following way for each UAV p_j :

$$\begin{aligned} q \in \mathcal{B}_j \quad & M(q, t) = \beta M(q, t) \\ q \in \mathcal{Q} \quad & M(q, t) = M(q, t) + \delta \end{aligned} \quad (3)$$

where $\delta > 0$ can be considered a forgetting factor with respect to measurements that have been taken in the past. Collision avoidance between UAVs is guaranteed by the construction of the Voronoi partitions. The reader is directed to [10] for details.

From design, the control algorithm is executed at the fusion center and communicates with all UAVs. The modified algorithm of [10] is outlined in Fig. 4.

```

1: while  $t < T_{final}$ ,  $IsDetected = \text{false}$  and  $OverWrite = \text{false}$  do
2:   for all UAVs  $p_j$ ,  $j = 1, \dots, n$  do
3:     if  $p_j = g_{pj}, \forall j = 1, \dots, n$  then
4:       Determine the Voronoi partition  $V_j$  in  $\mathcal{Q}$ .
5:       Apply the general Monte Carlo optimization method to determine an approximate maximum  $\tilde{g}^*$  in  $V_i$  of  $M(q, t)$ .
6:       Create a navigation function,  $f_j(p_j, g_{pj}, d)$  in  $V_j$  with  $g_{pj}$  set as the goal point.
7:       Create list of waypoints  $[wp_1, wp_2, \dots, wp_k]$  for UAV  $j$  by following  $-\nabla f_j(\cdot)$ .
8:     end if
9:     UAV  $j$  communicates with the fusion center and exchanges position information.
10:    if  $[x_T, y_T] \in \mathcal{B}_j$  then
11:       $IsDetected = \text{true}$ 
12:    end if
13:    For all points visited by UAV  $j$  and that are inside the FOV  $\mathcal{B}_j$ , update  $M(q, t)$  as in equation (3).
14:  end for
15: end while

```

FIG. 4 Prioritized search algorithm employed by the UAV team to search the most likely areas containing a target of interest.

B. ADAPTIVE SAMPLING-BASED FILTER FOR TARGET TRACKING

The target in Eq. (2) can be considered as a generalization of the Reeds–Shepp car. A difference is that, with zero linear velocity, the target can rotate while being at the same position. More generally, a turning radius $r(t)$ is not fixed but is a function of linear and angular velocity [$r(t) = v(t)/\omega(t)$]. The shortest path of the Reeds–Shepp car consists of circles and lines [5]. Therefore, the target’s path can be approximated by circles where lines are circles with infinite radius. Using the method explained in [11], an algorithm for fitting a circle to a number of given points is implemented.

UAV’s estimation of the target’s future position is based on fitting circles. Our algorithm considers only Γ detections, where $\Gamma \geq 3$ is a natural number, in calculating a fitting circle. Note that detections can come from different UAVs. Also, the target’s velocity is being estimated from the same set of detections. In other words, the UAVs are estimating both target velocity and position, while the target’s orientation is given as tangents to corresponding fitting circles.

A detection can be obtained only if a target is inside a UAV’s FOV and at time instances that are multiples of T_s . Idle UAVs make detections of targets entering their FOVs based on changes occurring in its “escape area” (9) on every multiple of T_s .

Because we are implementing intelligent pursuers (i.e., rational decision makers), a game theory approach is taken. Based on the current detection and the knowledge of the maximum linear velocity of a target, a UAV calculates the time needed for that target to leave its FOV using expression (5) and $\max\{|v_{\min}|, |v_{\max}|\}$, where $||$ is the absolute value. A UAV makes a new detection based on the time calculated. This is what we refer to as the adaptive filter.

More formally, the target can be seen as a Markov process \mathcal{T} given Eq. (2) where its position is being measured by a UAV p_j modeled as

$$y_j(kT_s) = H_j[x(kT_s)] + v_j(kT_s) \quad (4)$$

where $k \in \{0, 1, 2, \dots\}$. Function H_j is highly nonlinear because the sensor’s FOV is limited. From Eq. (4), it is obvious that measurements can be taken only at multiples of T_s . The measurement noise in Eq. (4) is additive and is denoted by $v_j(kT_s)$. The only necessary assumption on the noise is to be white with a finite variance. Instead of kT_s , we will use simply k for denoting sampling time. Because our filter does not require availability of input values, process noise is not needed.

After obtaining a new detection at time k , a minimum possible distance $d_j(m^*)$ at time m^* of \mathcal{T} to $\partial\mathcal{B}_j(m^*)$, where $\partial\mathcal{B}_j(m^*)$ is a boundary of FOV of p_j at time instant m^* , is given by

$$d_j(m^*) = \inf_{b \in \partial\mathcal{B}_j(m^*), \mathcal{A} \in \{\mathcal{B}_{\mathcal{T}}|y_j(k)\}} \{||\mathcal{A} - b||\} \quad (5)$$

where $||$ represents the Euclidean norm, \mathcal{A} represents the target’s position, and $\{\mathcal{B}_{\mathcal{T}}|y_j(k)\}$ is a set of points in \mathbb{R}^2 reachable by the target at instant m^* given the detection $y_j(k)$, $m^* \geq k$, and $m^* \in \mathbb{N}$.

At any time instant k , the expectation of an estimate of a position of \mathcal{T} is given as the expectation of a conditional expectation:

$$E[x(k)] = E[E[x(k)|\{y_{j_1}(k'_1), \dots, y_{j_\Gamma}(k'_\Gamma)\}]] \quad (6)$$

where $k'_i \leq k$. Based on the accuracy of an estimate, the set $\{k'_1, \dots, k'_\Gamma\}$ does not have to include the newest Γ detections. Expression (6) is calculated using a geometric approach. Note that our design allows any filter for this calculation (e.g., unscented Kalman filter or particle filter). However, in [12] we provide details and reasons in favor of choosing the geometric approach. Influence of measurement noise v_j is averaged out using $\Gamma > 3$ because a circle is uniquely defined with three points. Furthermore, a current velocity of the target is estimated based on Γ detections of the target. Velocities are calculated for intervals between two consecutive observation, and a least-mean-square (LMS) line is calculated.

A tradeoff between energy consumption for motion on the one side and the energy consumption for sensing and processing on the other side is also taken into account when designing our adaptive filter. A user can put its own preferences and account for noise in the developed algorithm by adjusting the size of the following areas of a FOV. “No moving area” is a central area (around a sensor’s platform) of a FOV such that, if a detection is obtained within it, the pursuer will not move towards the observed target. With the size of “no moving area,” comprised in η , we control how much the pursuer moves while tracking. If sensing and processing of gathered information require high energy consumption relative to energy consumption when moving, the size of “no moving area” is decreased to accommodate for that. When a detection of the targets position y_j is obtained by p_j at time instant k outside of its “no moving area,” the following control policy is applied until the next detection:

$$u_j(t) = K[y_j(k) - p_j] \quad (7)$$

where $K > 0$ is a proportional gain.

“Escape area” is an area closest to the boundary of a FOV. The “escape area” is larger (larger ζ) if the measurement noise level is higher (i.e., greater variance of the noise) in order to prevent a target to escape from the FOV. Therefore, based on the previous detection, a new detection is obtained when there is a chance for a tracked target to enter “escape area” in order to slip out from the FOV. “Escape area” also accounts for increased uncertainty of sensors towards the boundary of the FOV. These areas can differ between UAVs. Definitions of the “no moving area” and “escape area” for the j th UAV are

$$\mathcal{B}_j^{NM}(k) = \{q \in \mathcal{B}_j(k) : \|q - p_j\| \leq \eta_j\} \quad (8)$$

$$\mathcal{B}_j^{EA}(k) = \{q \in \mathcal{B}_j(k) : \inf_{b \in \partial \mathcal{B}_j(k)} \|q - b\| \leq \zeta_j\} \quad (9)$$

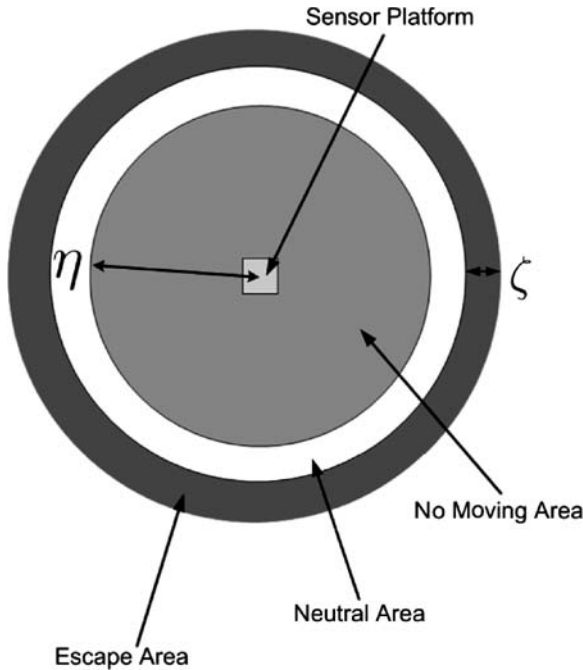


FIG. 5 Parts of a FOV of an omnidirectional sensor.

where η_j and ζ_j are user-determined parameters. An illustration of the FOV areas for an omnidirectional sensor is given in Fig. 5. The “neutral area” may not exist depending on the user’s preferences (i.e., sizes of “no moving area” and “escape area”) and does not have special functionality.

Now we have everything to define an adaptive filter’s decision whether to make a new observation in the next available time instant. The decision $\delta_j(k+1)$ is a nonanalytic $\{0, 1\}$ -valued function given with the algorithm in Fig. 6.

C. HUMAN-MACHINE INTERFACE

We propose a search mode in which the human selects waypoints of interest and the vehicles implement the appropriate low-level control as described in Sec. III.A to achieve desired positioning around those waypoints. One particularly important issue for this mode of operation is the choice of information to be displayed to the human. Consider a variety of displays, with an assortment of information about each vehicle. In many complex systems, it is a priority to determine the minimal information required to display on the user interface, not only because of limitations in the physical size of the display, but also because when too

```

1: Set parameters  $\Gamma$ ,  $\rho_j$ ,  $\eta_j$  and  $\zeta_j$  for all UAVs
2: Obtain sensors' readings and make a pair target-pursuer
3: For all pairs set  $\delta_j(index) = 1$ ,  $index \in \{1, 2\}$ 
4: while  $t \leq T_{final}$  and  $OverWrite = \text{false}$  do
5:   for all UAVs do
6:     if  $p_j$  is idle then
7:       Scan  $\mathcal{B}_j^{EA}(k)$  for new targets
8:       Exchange obtained information with fusion center
9:       if  $IsDetected = \text{true}$  then
10:        Start tracking
11:        Set  $\delta_j(index) = 1$ ,  $index \in \{k+1, k+2\}$ 
12:       end if
13:     else
14:       if  $\delta_j(k) = 1$  then
15:        Use corresponding sensor to scan FOV
16:        Exchange information (including  $y_j(k)$ ) with fusion center
17:        if  $\|E[x] - y_j(k)\| \geq \rho_j$  then
18:         Recalculate a fitting circle using last  $\Gamma$  detections
19:         if  $\|E[x] - y_j(k)\| \geq \rho_j$  then
20:          Set  $\delta_j(k+1) = 1$ 
21:         end if
22:        end if
23:        if  $y_j(k) \notin \mathcal{B}_j^{NM}(k)$  then
24:         Apply  $u_j(t)$  given in (7)
25:        end if
26:        if  $d_j(k) \leq \zeta_j$  then
27:         Set  $\delta_j(k+1) = 1$ 
28:        end if
29:        Make prediction  $E[x(k+1)]$ 
30:      end if
31:    end if
32:  end for
33:   $t = t + T_s$ , i.e.,  $k = k + 1$ 
34: end while

```

FIG. 6 Adaptive sampling rate algorithm for multi-agent tracking scenarios.

much information is presented, the human may simply ignore the displayed information altogether [13].

We presume that the human is a special type of observer, with restrictions that reflect the limits of human information processing as well as best practices in user-centered design. For example, in contrast to a standard observer, that relies on a history of observations and inputs to reconstruct the current value of the state of the system, a human observer may not have access to measurements of automation inputs and should also be able to reconstruct the state of the system based on current observations only. Hence, we use necessary conditions, derived in [14] and based on unknown-input observability, that can be used to evaluate whether a given user interface will allow the human observer to reconstruct information about the system relevant to a particular task (also formulated as a linear combination of system states). We posit that displays that satisfy our modified

unknown-input observability conditions contain at least the minimum information required for effective human-automation interaction.

We consider a high-level model of formation dynamics based on [15] that incorporates a controller for formation stabilization, in which the fleet of vehicles must track human-chosen waypoints. For N point-mass vehicles with states $q_i \in \mathbb{R}^n$, $i \in \{1, \dots, N\}$ and communication topology as represented by the graph Laplacian L , we have dynamics

$$\dot{q} = (A + BFL)(q - h) + Bu_{\text{track}} + BNr \quad (10)$$

with formation stabilization controller $u = FL(q - h)$ for stabilization to known formation h and tracking controller $u_{\text{track}} + Nr$, with user-selected reference input r that represents the desired waypoints. Each vehicle uses measurements that it receives from its neighbors to stabilize itself to formation via the control gain F , chosen as in [15] for L . N is chosen as per standard methods [16] to remove steady-state tracking error. We consider an output that represents the centroid of the vehicle fleet,

$$y = \frac{1}{N} \mathbf{1}^T q \quad (11)$$

and seek to determine whether a task that involves positioning of the fleet around a desired waypoint is observable and predictable by the user with this output. On a visual display (e.g., a monitor or tablet that shows a map of the search area, for example) this output is consistent with a single marker indicating position of the centroid of the fleet.

As in [14], we compute the user-observable subspace \mathcal{O}_H and the user-predictable subspace \mathcal{P}_H (which is a subset of the user-observable subspace),

$$\mathcal{O}_H = \mathcal{R}(C^T) \oplus A^T \sum_{i=0}^{n-2} \left\{ \mathcal{R}[(A^i)^T C^T] \cap \mathcal{N}(B_{\text{both}}^T) \right\} \quad (12)$$

$$\mathcal{P}_H = E_{\mathcal{O}_H} [\mathcal{N}(\bar{A}_{12}^T) \cap \mathcal{N}(\bar{B}_{a\mathcal{O}_H}^T)]$$

where $\mathcal{R}()$ indicates the range space, $\mathcal{N}()$ indicates the null space, $E_{\mathcal{O}_H}$ is the basis of the user-observable subspace, and \bar{A}_{12} and $\bar{B}_{a\mathcal{O}_H}$ are the submatrices that arise in a transformation to user-observable form as in [14] and refer to portion of the transformed state and input matrix that are directly affected by the automation's input u_{track} . Unknown-input observability is used to determine those linear combinations of states that can be reconstructed at every instant despite not knowing the automation input u_{track} . The user-predictable subspace provides those linear combinations of states whose derivatives can additionally be reconstructed at every instant. If the task subspace, that is, the linear combination of states which are directly relevant for the task, is contained within the user-predictable subspace, we say that the user-interface meets the necessary requirements for a "good" display.

For Eq. (10) with a double-integrator model for each vehicle such that $A = I_{N \times N} \otimes A_{\text{veh}}$, $B = I_{N \times N} \otimes B_{\text{veh}}$, with graph Laplacian

$$L = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix} \quad (13)$$

corresponding to a bidirectional, fully connected topology for four vehicles, and formation controller $F = I_{N \times N} \otimes F_{\text{veh}}$ with $F_{\text{veh}} = [-1/8, -3/4]^T$, we find while the position and velocity of the centroid is observable, only the position of the centroid is user predictable. That is, because the user does not have information about the automated tracking controller, only position of the centroid is predicted (unlike velocity of the centroid, which is directly affected by the unknown input, and therefore cannot be predicted at any given instant). However, because the task involves direct knowledge only of the position, the output meets necessary conditions to ensure that the user has adequate information for the task.

If the human did have additional access to the tracking control law, both the position and the velocity of the centroid would be user predictable. Also notice that this reduced output does not provide measurement of each individual vehicle, and hence, if the task required knowledge of the position of each individual vehicle, the display would be deemed problematic. Because the display was chosen to be minimal, however, and the task does not require knowledge of every vehicle's position individually, this analysis indicates that a reduced display, with only knowledge of the centroid of the fleet, would be adequate for positioning of the centroid at a desired waypoint.

IV. QUADROTOR BASELINE CONTROLLER DESIGN

A. QUADROTOR MODELING

The most common practice in flight control is to linearize the aircraft dynamics about different trim conditions and use gain-scheduled linear control techniques for aircraft control during different flight regimes [17, 18]. For increasingly stringent design requirements of multi-objective, highly constrained missions, such an approach may be inadequate [19]. In this section we present a nonlinear dynamic model of a quadrotor depicted in Fig. 7.

With some abuse of notation, the quadrotor dynamics are derived from first principles to describe a six-degrees-of-freedom rigid-body model driven by forces and moments [20, 21]. The reference frames used are shown in Fig. 7. The ground frame $\{G\}$ is the inertial frame fixed to the Earth. The aircraft frame $\{A\}$ is fixed to the moving aircraft and has its origin at the center of mass of the quadrotor. The transformation matrix between these reference frames is obtained by matrix multiplication of the three basic orthogonal rotation matrices that belong to a special orthogonal group $SO(3, \mathbb{R})$ [22].



FIG. 7 Hummingbird quadrotor with used inertial frames.

First, define the following variables to describe the aircraft dynamics: $\eta_1 = [x \ y \ z]^T$ (position of the origin of $\{A\}$ measured in $\{G\}$), $\eta_2 = [\phi \ \theta \ \psi]^T$ (angles of roll ϕ , pitch θ , and yaw ψ that parameterize locally the orientation of $\{A\}$ with respect to $\{G\}$), $v_1 = [u \ v \ w]^T$ [linear velocity of the origin of $\{A\}$ relative to $\{G\}$, expressed in $\{A\}$ (i.e., body-fixed linear velocity)], $v_2 = [p \ q \ r]^T$ (angular velocity of $\{B\}$ relative to $\{G\}$, expressed in $\{A\}$ i.e., body-fixed angular velocity). Table 1 shows the symbols used in quadrotor modeling.

The kinematics of the quadrotor model are given by

$$\begin{bmatrix} \dot{\eta}_1 \\ \dot{\eta}_2 \end{bmatrix} = \begin{bmatrix} {}^G_A R(\eta_2) & 0 \\ 0 & Q(\eta_2) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \iff \dot{\eta} = J(\eta)v, \quad (14)$$

with the matrix

$$Q(\eta_2) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \quad (15)$$

that is singular for $\theta = \pm \pi/2$. This singularity will not represent a problem in our design because the quadrotor will not execute aggressive maneuvers as to achieve the pitch of 90 deg. To include aggressive maneuvers, the singularity

TABLE 1 NOTATION SYMBOLS USED FOR QUADROTOR MODELING

DOF	Motion	Forces	Velocities
1	x direction	F_x	u
2	y direction	F_y	v
3	z direction	F_z	w
4	ϕ (roll)	T_ϕ	p
5	θ (pitch)	T_θ	q
6	ψ (yaw)	T_ψ	x

can be circumvented by various methods, such as switching to quaternion representation.

The dynamics consists of equations for translation and rotation, which are represented with Newton–Euler’s equations. In expanded form, the nonlinear quadrotor model is written as

$$\begin{aligned}
 m(\dot{u} - vr + wq) &= F_x \\
 m(\dot{v} - wp + ur) &= F_y \\
 m(\dot{w} - uq + vp) &= F_z \\
 I_{xx}\dot{p} + (I_{zz} - I_{yy})qr &= T_\phi \\
 I_{yy}\dot{q} + (I_{xx} - I_{zz})rp &= T_\theta \\
 I_{zz}\dot{r} + (I_{yy} - I_{xx})pq &= T_\psi
 \end{aligned} \tag{16}$$

Forces and torques are generated as

$$f_\tau(\eta_2) = {}^G_A R^{-1}(\eta_2) \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix}, \quad \tau(\eta_2, U) = \begin{bmatrix} f_\tau(\eta_2) \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

where U_1, U_2, U_3, U_4 represent control forces and torques generated by four quadrotor rotors [20].

B. LOW-LEVEL CONTROLLER DESIGN

We use feedback linearization [23] to stabilize the model in Eqs. (14) and (16), which can be represented in standard affine form:

$$\begin{aligned}
 \dot{q} &= f(q) + g(q)u \\
 y &= Cq
 \end{aligned} \tag{17}$$

with state $q = [\eta_1 \ \eta_2 \ v_1 \ v_2]^T \in \mathbb{R}^{12}$, input $u = [U_1 \ U_2 \ U_3 \ U_4] \in \mathbb{R}^4$, and output $y = [\eta_1 \ \eta_2]^T \in \mathbb{R}^6$. Because the quadrotor model is a nonlinear underactuated system, we exploit the differential flatness property that allows all states and inputs to be expressed as functions of the outputs and a finite number of their derivatives [24]. Moreover, these systems have a useful property that there is a one-to-one mapping between trajectories in output space and trajectories in the state and input spaces. Taking this into account, for the system (17) we chose four flat outputs as $[\eta_1(3) \ \eta_2]^T = [z \ \phi \ \theta \ \psi]^T \in \mathbb{R}^4$, which allows us to proceed using the feedback linearization algorithm for square multi-input multi-output (MIMO) system.

The system meets existence conditions [23] (Lemma 5.2.1) for well-defined relative degree $r = 2 + 2 + 2 + 2 = 8$ at a point $q^0 \in \mathcal{R}$ [12]. Because the matrix

$$A(q) = \begin{bmatrix} L_{g1}L_f^3h_1(q) \dots L_{g4}L_f^3h_1(q) \\ \vdots \\ L_{g1}L_f^3h_4(q) \dots L_{g4}L_f^3h_4(q) \end{bmatrix} \quad (18)$$

is nonsingular at $q = q^0$, we use exact state feedback to partially linearize the system with controller

$$u = A(q)^{-1}[-b(q) + \vartheta] \quad (19)$$

with $b(q) = [L_f^{r_1} \ L_f^{r_2} \ L_f^{r_3} \ L_f^{r_4}]^T$ and external input ϑ chosen to track a reference trajectory y_r by stabilizing error dynamics $\dot{e} = \dot{y} - \dot{y}_r$.

The zero dynamics are

$$\begin{aligned} \dot{\xi}_1 &= \xi_2 \\ \dot{\xi}_2 &= -c_1 \xi_2 \\ \dot{\xi}_3 &= x_4 \\ \dot{\xi}_4 &= -c_2 \xi_4 \end{aligned} \quad (20)$$

where $\xi \in \mathbb{R}^4$ and c_1 and c_2 are positive constants. Because the zero dynamics in this case are linear, their analysis is trivial. Because each of the two separate subsystems of Eq. (20) have one zero eigenvalue and the other eigenvalue with a negative real part, the zero dynamics are stable. Further, because Eq. (20) has a nontrivial null space, the system has an equilibrium subspace, rather than an equilibrium point. Physically, the zero dynamics correspond to positions x , y and their linear velocities u , v , respectively. It is obvious that these states

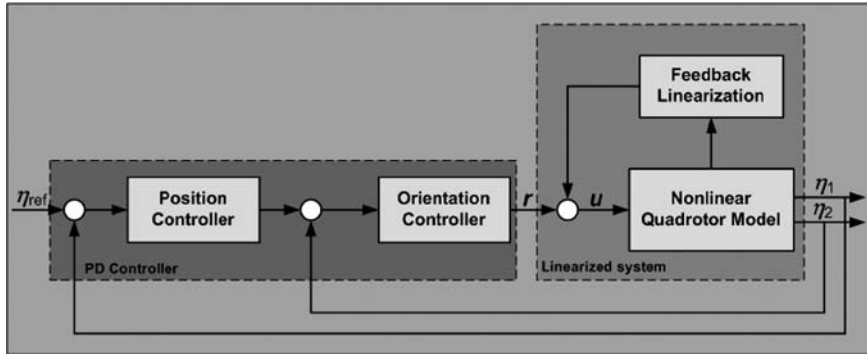


FIG. 8 Cascade PD controller.

are not controllable directly through the input vector $[U_1 \ U_2 \ U_3 \ U_4]^T$ because $F_x(U_1, \phi, \theta, \psi), F_y(U_1, \phi, \theta, \psi)$ are such that $F_x(U_1, 0, 0, 0) = 0$ and $F_y(U_1, 0, 0, 0) = 0$. Furthermore, the equilibrium subspace represents stable hovering of the quadrotor in every point in two-dimensional Cartesian space with linear velocities equal to zero $u = 0, v = 0$. Because the zero dynamics are stable, we are able to design the controller to stabilize and control the nonlinear quadrotor model. The proposed controller consists of two parts: the first part is nonlinear, which linearizes the system through output feedback linearization; the second part of the controller is linear, and its purpose is to stabilize and control all six DOF of the quadrotor, position η_1 , and orientation η_2 , through a cascade structure as illustrated in Fig. 8.

V. SIMULATIONS AND EXPERIMENTS

A. SIMULATIONS

To simulate our hybrid framework for user-guide prioritized search, we assume a rectangular AOI (30×30 m) with four UAVs having FOVs of 1.5, 2.0, 2.1, and 2.5 m. Maximum target linear and angular velocities are set to 0.7 m/s and 1.5 rad/s respectively with UAV maximum velocity set at 5 m/s. The probability of detection map parameters β and δ are set to 0.4 and 5×10^{-3} . In Fig. 9 we see that three areas have high probabilities of containing the target of interest initially. The UAVs are shown by the ovals as well as their FOVs. The PTS algorithm is implemented to identify points in the AOI that have the highest probability of containing the target of interest. Waypoint information is relayed to the onboard low-level controller to navigate the UAVs to those regions for searching.

When the target of interest is detected by one of the UAVs, their behavior changes from the PTS algorithm to the AFTT algorithm for target tracking.

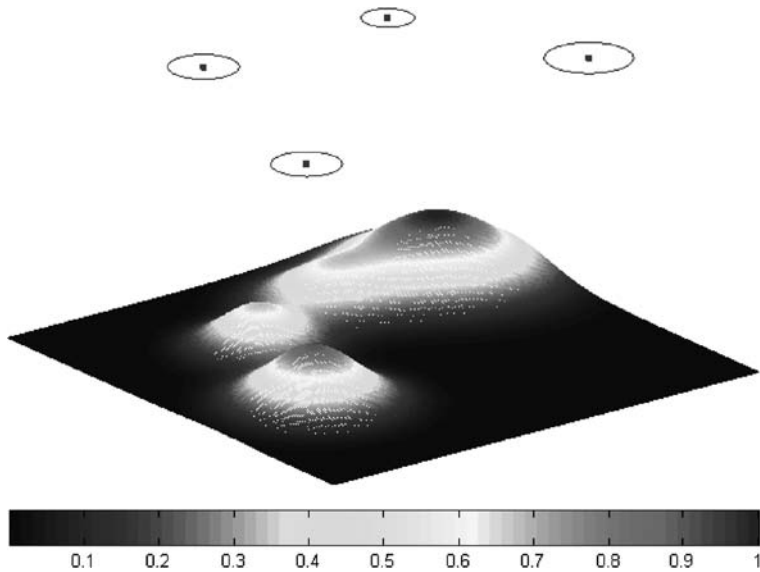


FIG. 9 Initial probability of detection map. UAVs and their respective FOVs are shown in upper ovals.

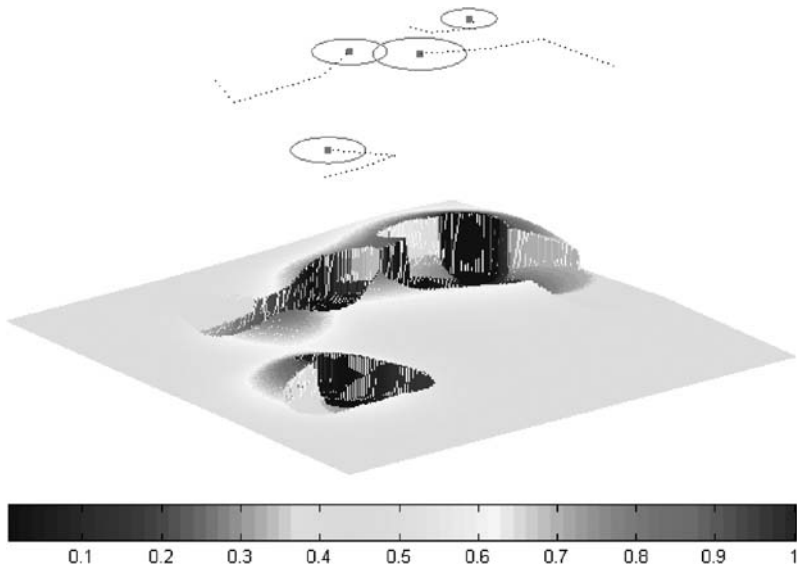


FIG. 10 Reduced probability of detection map at the moment the UAVs switch from the searching algorithm (PTS) to the target tracking algorithm (AFTT). UAVs are shown in upper ovals, and their trajectories are shown by dotted black lines.

Figure 10 shows the reduction of the probability of detection map when the UAVs switch from the PTS algorithm to the AFTT algorithm as well as their respective trajectories while implementing the PTS algorithm.

Figure 11 shows a snapshot of the AFTT algorithm. The target of interest is shown as an arrow, and its trajectory is the dashed line. The dotted line shows the fitting circle used by the AFTT algorithm. Notice that the fitting circle is a line, indicating a circle with almost infinite radius. The magenta hashed line shows the tracking UAV's trajectory. During this simulation only one UAV was actively tracking; however, the AFTT algorithm allows for “handing off” the target to other UAVs if the target enters their FOV.

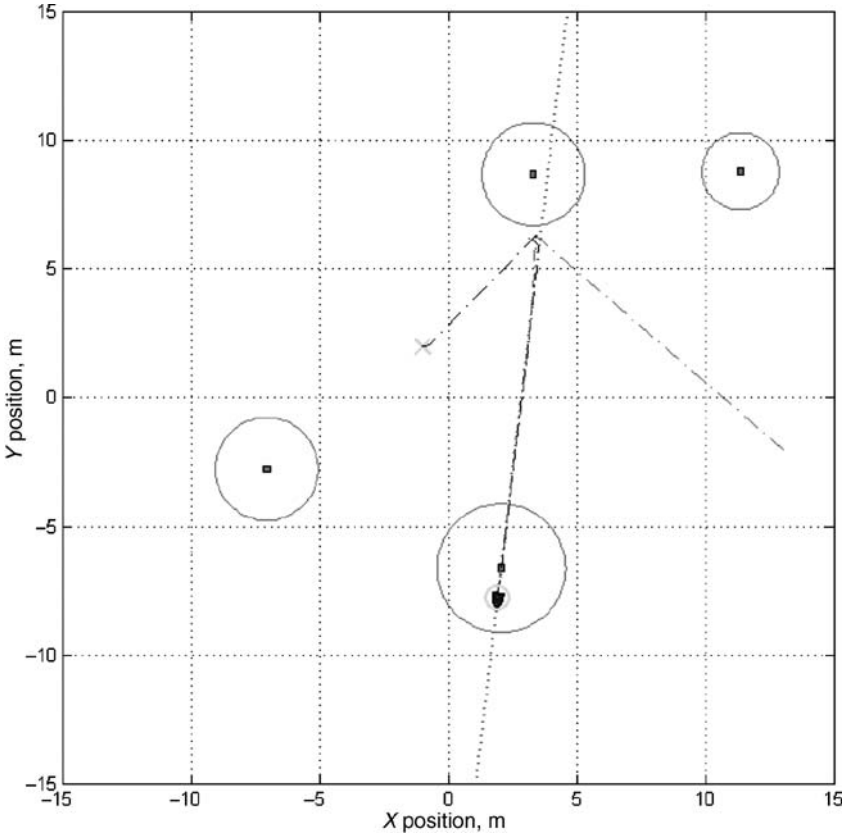


FIG. 11 Snapshot of the tracking algorithm. The target is seen as an arrow. The fitting circle from AFTT algorithm is shown as a dotted line. The dashed lines are the target trajectory and the trajectory of the UAV that is currently tracking.

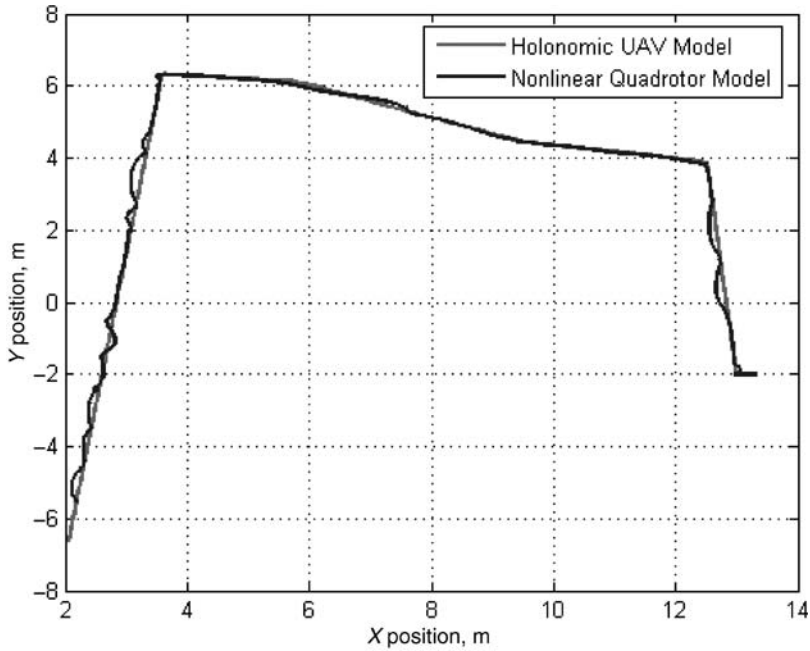


FIG. 12 Comparison of waypoint tracking for holonomic UAV model and nonlinear quadrotor model.

Comparison of waypoint tracking between a holonomic UAV model and the dynamic quadrotor model derived in Sec. IV.A is shown in Fig. 12. The trajectories of the holonomic and dynamic quadrotor model are very similar, indicating the low-level controller does a good job of steering the quadrotors to the desired waypoints. Figures 13 and 14 show the dependence of quadrotor velocity to the distance from the desired waypoint. The velocity is high when a new waypoint is received and then decreases as the quadrotor approaches the desired waypoint.

B. HARDWARE EXPERIMENTS

Hardware experiments to qualitatively verify the target search and tracking framework provided are carried out at the MARHES (Multi-Agent Robotics Hybrid and Embedded Systems) laboratory at the University of New Mexico (UNM). A heterogeneous multivehicle testbed consists of five Pioneer P3-AT mobile robots, ten TXT-1 nonholonomic car-like vehicles, three AscTech Hummingbirds quadrotors (data available online at <http://www.asctec.de/>),

and other commercially available robotic vehicles. In addition, our sensor suite is made of Hokuyo and Sick laser range finders for obstacles avoidance, Garmin GPS, Microstrain IMUs, stereo cameras, Kinect sensors, and XBee ZigBee radios for networking. Also a Vicon motion capturing system (data available online at <http://www.vicon.com>) composed of eight cameras is installed for precision indoor positioning.

Figure 15 shows a block diagram of how the testbed is connected during experiments indoor and outdoor. Figure 15 focuses primarily on the communications aspect. ZigBee and Wi-Fi routers are the main means of communication used in our experiments. The NI CompactRIO (data available online at <http://www.ni.com/compactrio/>), interfaced with the ZigBee RF module, creates the communication bridge between a base station and a quadrotor.

Figure 16 shows one of the quadrotor hovering above one of the Pioneer ground vehicles, overseen by the Vicon system (two cameras visible in the figure), and in the background two of the TXT-1 platforms together with more Pioneer robots and other vehicles.

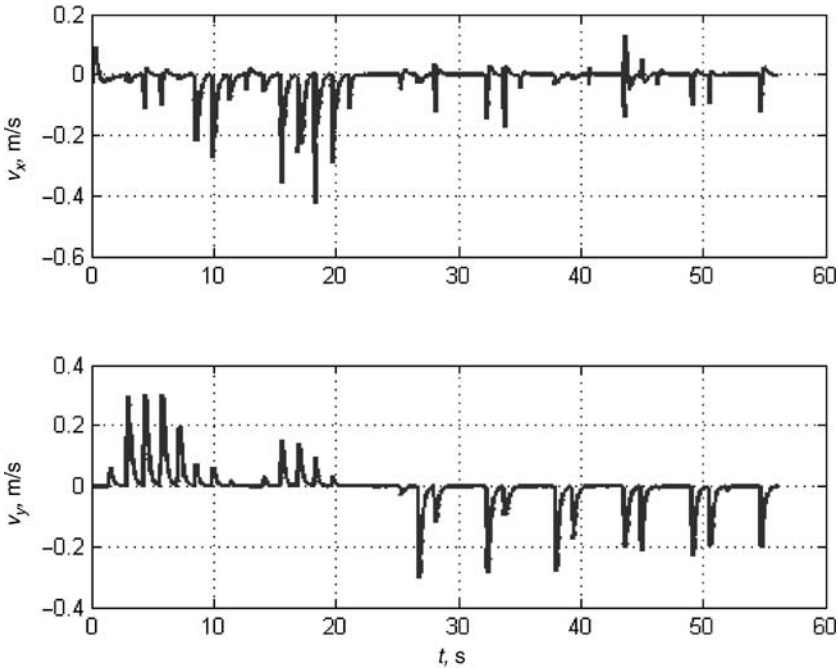


FIG. 13 Velocities of nonlinear quadrotor model (x and y components) during waypoint tracking.

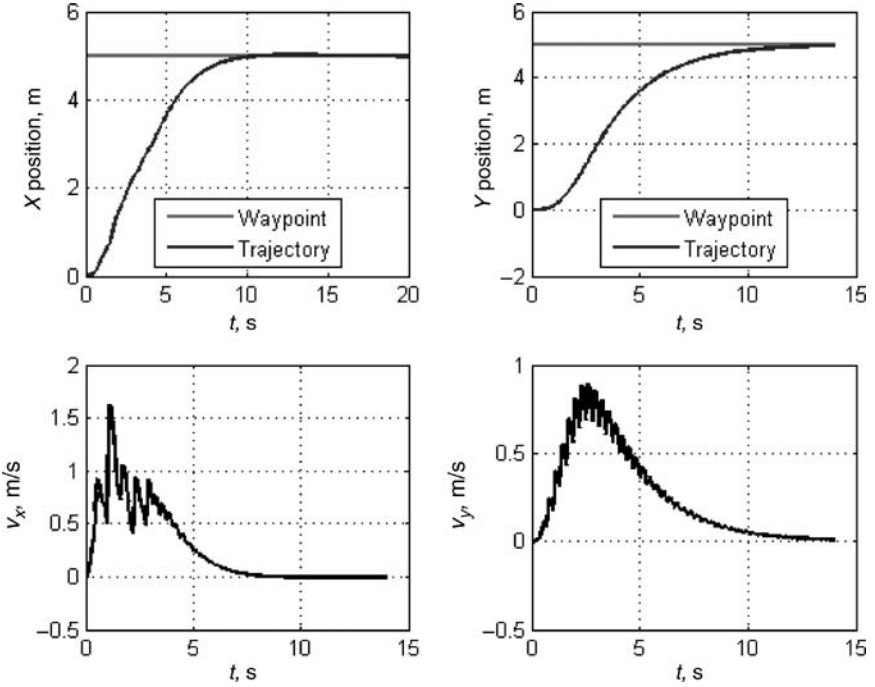


FIG. 14 Quadrotor velocity profile for a single waypoint. Both x and y components are shown.

The data provided from the Vicon motion capture system are used to localize the quadrotors as well as to simulate a virtual sensor with a disk shape FOV for each quadrotor. The fusion center will compute y_i with the Vicon motion capture system information. Although this is a slight modification to the global scheme described in Sec. II.B, the structure of the framework remains intact.

VI. CONCLUSIONS AND FUTURE WORK

In this chapter we present a framework that combines high-level motion planning and low-level real-time control for a team of UAVs tasked with searching and tracking a target of interest. The framework allows for human-in-the-loop scenarios when factors such as human intuition can be exploited by the UAV team. Our framework also addresses two main elements that are often overlooked in the literature, namely, vehicle dynamics and human-in-the-loop. Simulations show the ability of the framework to locate and subsequently track a target of interest.

Future work entails extending the search/tracking algorithms to multiple targets of interest as well as trying to relax assumptions on connectivity and same altitude flight. Also a robustness analysis of our low-level controller needs to be conducted to understand its effectiveness in the presence of noisy measurements and disturbances.

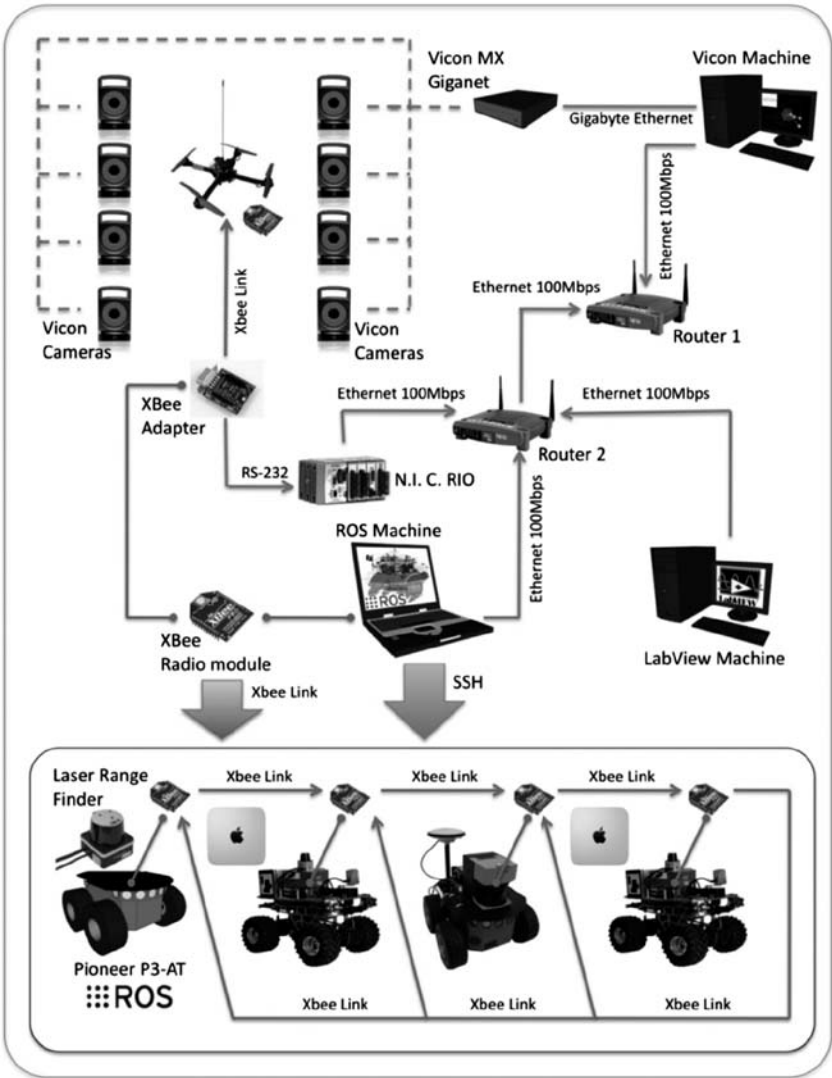


FIG. 15 Block diagram with testbed and communication links for experimental purposes.



FIG. 16 MARHES testbed with a quadrotor aerial vehicle hovering over one of the Pioneer P3-AT robots, overseen by a motion capturing system.

REFERENCES

- [1] Lavis, B., Furukawa, T., and Durrant-Whyte, H. F., "Dynamic Space Reconfiguration for Bayesian Search and Tracking with Moving Targets," *Autonomous Robots*, Vol. 24, No. 4, 2008, pp. 387–399.
- [2] Tisdale, J., Kim, Z., and Hedric, K., "Autonomous UAV Path Planning and Estimation," *IEEE Robotics and Automation Magazine*, Vol. 16, No. 2, 2009, pp. 35–42.
- [3] Hoffmann, G. M., and Tomlin, C. J., "Mobile Sensor Network Control Using Mutual Information Methods and Particle Filters," *IEEE Transactions on Automatic Control*, Vol. 55, No. 1, 2010, pp. 32–47.
- [4] Tang, Z., "Information-Theoretic Management of Mobile Sensor Agents," Ph.D. Dissertation, Ohio State Univ., Columbus, OH, 2005.
- [5] LaValle, S., *Planning Algorithms*, Cambridge Univ. Press, 2006, pp. 437–495.
- [6] Gelman, A., "Objections to Bayesian Statistics," *Bayesian Analyses*, Vol. 3, No. 3, 2008, pp. 445–450.
- [7] Zengin, U., and Dogan, A., "Real-Time Target Tracking for Autonomous UAVs in Adversarial Environment: A Gradient Search Algorithm," *IEEE Transactions on Robotics*, Vol. 23, No. 2, 2007, pp. 294–307.

- [8] How, J. P., Cameron Fraser, K. C. K., Bertuccelli, L. F., Toupet, O., Brunet, L., Bacharach, A., and Roy, N., "Increasing Autonomy of UAVs," *IEEE Robotics and Automation Magazine*, Vol. 16, No. 2, 2009, pp. 43–51.
- [9] How, J., Fraser, C., Kulling, K., and Bertuccelli, L., "Increasing Autonomy of UAVs," *IEEE Robotics and Automation Magazine*, Vol. 16, No. 2, 2009, pp. 43–51.
- [10] Cortez, R. A., Fierro, R., and Wood, J., "Prioritized Sensor Detection via Dynamic Voronoi-Based Navigation," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 5815–5820.
- [11] Pratt, V., "Direct Least-Squares Fitting of Algebraic Surfaces," *ACM SIGGRAPH Computer Graphics*, Vol. 21, No. 4, 1987, pp. 145–152.
- [12] Tolić, D., and Fierro, R., "Adaptive Sampling for Tracking in Pursuit-Evasion Games," *2011 IEEE Multi-Conference on Systems and Control (MSC 2011)*, 2011, pp. 179–184.
- [13] Wickens, C., and Hollands, J., *Engineering Psychology and Human Performance*, Prentice–Hall, Upper Saddle River, NJ, 2000.
- [14] Eskandari, N., and Oishi, M., "Computing Observable and Predictable Subspaces to Evaluate User-Interfaces of LTI Systems Under Shared Control," *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, Best Student Paper Award, Oct. 2011.
- [15] Williams, A., Lafferriere, G., and Veerman, J., "Stable Motions of Vehicle Formations," *Proceedings of the IEEE Conference on Decision and Control*, Dec. 2005, pp. 72–77.
- [16] Fax, J., and Murray, R., "Information Flow and Cooperative Control of Vehicle Formations," *IEEE Transactions on Automatic Control*, Vol. 49, No. 9, Sept. 2004, pp. 1465–1476.
- [17] Sutarto, H. Y., Budiyo, A., Joelianto, E., and Hiong, G. T., "Switched Linear Control of a Model Helicopter," *International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2006, pp. 1–8.
- [18] Gonzalez, A., Mahtani, R., Bejar, M., and Ollero, A., "Control and Stability Analysis of an Autonomous Helicopter," *Proceedings of World Automation Congress*, Vol. 15, June 2004, pp. 399–404.
- [19] Michael, N., Fink, J., and Kumar, V., "Cooperative Manipulation and Transportation with Aerial Robots," *Proceedings of Robotics: Science and Systems*, June 2009.
- [20] Bouabdallah, R., "Backstepping and Sliding-Mode Techniques Applied to an Indoor Micro Quadrotor," *IEEE International Conference on Robotics and Automation*, April 2005.
- [21] Fossen, T. I., *Guidance and Control of Ocean Vehicles*, Wiley, New York, 1999.
- [22] Arfken, G., *Mathematical Methods for Physicists*, 3rd ed., Academic International Press, New York, 1985.
- [23] Isidori, A., *Nonlinear Control Systems*, 3rd ed., Springer-Verlag, Berlin, 2001.
- [24] Fliess, M., Levine, J., Martin, P., and Rouchon, P., "Flatness and Defect of Nonlinear Systems: Introductory Theory and Examples," CAS Internal Report A-284, France, Jan. 1994.

SUPPORTING MATERIALS

Many of the topics introduced in this book are discussed in more detail in other AIAA publications. For a complete listing of titles in the Progress in Astronautics and Aeronautics series, as well as other AIAA publications, please visit www.aiaa.org.

AIAA is committed to devoting resources to the education of both practicing and future aerospace professionals. In 1996, the AIAA Foundation was founded. Its programs enhance scientific literacy and advance the arts and sciences of aerospace. For more information, please visit www.aiaafoundation.org.

INDEX

Note: Page numbers with “f” represent figures. Page numbers with “t” represent tables. Page numbers with “L” represent listings.

Index Terms

Links

A

AANN. *See* Auto-associative neural network.

AC. *See* Adaptive critic.

ACK. *See* Acknowledgment.

Acknowledgment (ACK) 415

Action network 34 60

Actors 390–391

Actuators 226

actuation system 246–247

mechanical design 29

SISO 218

VBV and VSV 208

for wing dihedral 24

Adaptive critic (AC) 60

formulations 60

for helicopter optimal control 43

model-based synthesis 61

Adaptive sampling-based filter for target

tracking (AFTT) 449

See also Unmanned aerial vehicle

(UAV).

escape area 452

This page has been reformatted by Knovel to provide easier navigation.

Index Terms

Links

Adaptive sampling-based filter for target
tracking (AFTT) (*Cont.*)

FOV omnidirectional sensor parts	453f
Markov process	451
minimum possible distance	451
neutral area	453
no moving area	452
reduced probability of detection	
map	461f
target detections	451

ADP. *See* Approximate dynamic
programming.

Advanced Health Management System
(AHMS) 176–177

Aerodynamics

See also Micro aerial vehicle (MAV).

frame rotations	9
Goman and Khrabrov's model	9–10
inertial frames	8
lift and quarter-chord moment	10
pendulum rig	8
perching	3
pitch rotation point	7
Quanser pendulum platform	7
unsteady flow conditions	10
wing kinematics	7–8

AFTT. *See* Adaptive sampling-based filter
for target tracking.

Agent-oriented software engineering
(AOSE) 395 397

Index Terms

Links

AHM. *See* Airplane Health Management.

AHMS. *See* Advanced Health
Management System.

Air-data vane calibration 159

See also Flight control law, L_1 adaptive.

angle-of-attack vane calibration 159

using constant strategy 160f 161f 162f

using variable strategy 160f

angle-of-sideslip vane calibration 159f

flat turn 163f 164f

flight 162f

Airplane Health Management

(AHM) 176

Amigo-bot hardware platform 429 430f

Animal flight 1

analysis by Floquet theory 2

flight regimes 1 3

robotic flapping flyers 2

AOI. *See* Area of interest

AOSE. *See* Agent-oriented software
engineering.

Approximate dynamic programming

(ADP) 63

formulation 60

infinite time problem 64

optimal control steps 64–65

Area of interest (AOI) 447 448

Astronaut–robot interaction 388

Auto-associative neural network

(AANN) 210

Index Terms

Links

Autonomous robot teams

See also Robot agent.

motivating scenario	411	412–413
notional lunar area	412f	
protocol development	413	
astronaut initiated message		
exchange	415t	
complex protocol	414	415
iterative process features	414t	
robot-to-robot team messaging		
activity	415t	
TWE protocol	413	
recharging protocol	419	
phases	419	
thread of message exchanges	420	
robot-initiated safety protocol		
proximity alert protocol	422	423
proximity avoid protocol	422	
proximity safing protocol	419	421–422
TWE protocol	416	
astronaut oversight	417	418t
finite state machine view	416f	
sequences	416	417
simpler protocols	417t	
team leader robot	418–419	
worker robot agents	418	

Autonomous soaring updraft/thermal

model	298
dimensionality	302
multiple thermals in	

Index Terms

Links

Autonomous soaring updraft/thermal		
model (<i>Cont.</i>)		
two-dimensions	303f	
numerical dimensionality	303	
Monte Carlo simulation	303	
discretization levels	305	
using episodes	305–306	
multiple thermals in		
two-dimensions	304f	305f
PC applying	302	
policy analysis	306	
thermal learning parameters	302t	
thermal location casting	298–301	
value function	306	
Autonomy mode change protocol	400	
coordinator/lead agent role	400	401
for nonvirtual MAS	400	401f
for virtual MAS	402–403	

B

Backstepping technique	38	
Backstepping-based controller	34	
Bang-bang action	62	
BDI agents. <i>See</i> Belief-desire-intent agents.		
Belief	312	
Belief MDP problem	320	
Belief space	310	346
feedback policy	316–317	
FIRM	312	352

<u>Index Terms</u>	<u>Links</u>	
Belief space (<i>Cont.</i>)		
Gaussian	353	
MDP in	323	
Belief-desire-intent agents (BDI		
agents)	407	
Bellman equation	278	
Bellman optimality equations	279	
Biological fliers	2	
Burst-chop	207	
Bypass ratio	203	
C		
CAS. <i>See</i> Control augmentation system.		
Cascade PD controller	460f	
CBM. <i>See</i> Condition-based maintenance.		
CBM layered health DIaK architecture	181f	
Cell decomposition technique	312	
Central pattern generator (CPG)	2	4
<i>See also</i> RoboBat.		
actuator/structure technology	4	
central pattern generators	4	
Hopf oscillator	5	
neuron dynamics	4	
phase synchronization	6	
control design	5	
Hopf oscillator	5	
neuron dynamics	4	
phase synchronization	6	
Cerebellar model articulation controller		
(CMAC)	288	289

Index Terms

Links

CFD. <i>See</i> Computational fluid dynamics.			
Chemical steam generator (CSG)	192	193	
CHR. <i>See</i> Cooper–Harper ratings.			
CMAC. <i>See</i> Cerebellar model articulation controller.			
Cognitive manager	409		
Cognitive robot agents			
Amigo-bot hardware platform	429	430f	
worker and team leader robot simulation	426		
Computational fluid dynamics (CFD)	10		
Computed off-line	316	349	358
Condition-based maintenance (CBM)	178		
Control augmentation system (CAS)	142		
Control law design	16		
angle-of-attack control	16	17	
derivative-integral controller	18		
elevator deflection	17		
open-loop dynamics	18		
controller	16		
perching guidance loop	20		
simulations	16	17f	18f
yaw control by wing dihedral	19		
Control surface failures, L_1 adaptation in	138		
ten-deg locked-in-place left-aileron failure	140		

<u>Index Terms</u>	<u>Links</u>		
Control surface failures, L_1 adaptation (<i>Cont.</i>)			
two-deg locked-in-place left-aileron			
failure	139f		
Controller performance analysis	255		
characteristic parameters	258t		
fitness function value	257f		
FLC step responses	261f	262	
fuzzy controller	260f		
fuzzy membership functions	259f		
fuzzy system performance chart	257t		
genetic algorithm performance			
comparison	256t		
performance measure	258f		
Controller robustness	244		
Conversation manager	409		
Cooperative search, acquisition, and track			
(CSAT)	446		
Cooper–Harper ratings (CHR)	148	156	157t
Cost-function-based single network			
adaptive critic synthesis			
(J-SNAC synthesis)	62	63	
HJB equation with constraints	65–66		
neural network training	66–67		
with nonquadratic cost	65		
weight update law	85–87		
CPG. <i>See</i> Central pattern generator.			
Critic network	34–35	60	
CSAT. <i>See</i> Cooperative search,			
acquisition, and track.			
CSG. <i>See</i> Chemical steam generator.			

Index Terms

Links

Curse of dimensionality	281	310	
Cybele TM agent	407	429	
D			
DARE. <i>See</i> Discrete algebraic Riccati Equation.			
Data	174		
Data, information, and knowledge (DIaK)	175		
iISHM functions	175	177	
iISHM-DM	181	184	190
integrated management	185		
Data, information, and knowledge architecture (DIaKA)	182	183f	
CBM layered health	181f		
iISHM-DMs	185		
sensors and components	186		
DCF. <i>See</i> Distributed control framework.			
Decision making under uncertainty			
<i>See also</i> Motion planning problem.			
mathematical formulation	318–319		
MDP	319		
characterization	319		
DP equation	319–320		
GPRM	320		
POMDP	320		
belief MDP problem	320		
FIRM	321		
local controllers	322		
in robotic systems	345–346		

Index Terms

Links

Decision manager	410	411
Decision-maker	276	
Decision-making structures	410	
allowable values of state dimensions	410t	
decision modules	411f	
protocol-related foci	411	
Defuzzifier	234	
<i>See also</i> Fuzzifier		
Degree of intelligence (DoI)	175	
Degrees of freedom (DOF)	1	314
helicopter	33	
inverted pendulum system	288	289–290
rigid-body model	456	
testbed	7	
Departure-prone edge exploration	165	168
aircraft response	167f	
angle of attack <i>vs.</i> angle of sideslip		
coverage	166f	
Deployed robot team, protocol		
utilization by		
<i>See also</i> Simulated autonomous robot		
team effort, protocol		
utilization in.		
changes to robot agent entity	430	
laboratory environment	428	429f
physical environment elements	429	430
physical robot	429	
robotic software development		
process	431f	

<u>Index Terms</u>	<u>Links</u>		
Deployed robot team, protocol (<i>Cont.</i>)			
robots engaged in scientific sampling			
mission	431–438		
simulations	431		
Designated/designee role	397		
DHP. <i>See</i> Dual heuristic programming.			
DIaK. <i>See</i> Data, information, and			
knowledge.			
DIaKA. <i>See</i> Data, information, and			
knowledge architecture.			
Discrete algebraic Riccati equation			
(DARE)	356		
Distributed control framework (DCF)	407		
cognitive manager	409		
robot agent within	408f		
DOF. <i>See</i> Degrees of freedom.			
DoI. <i>See</i> Degree of intelligence.			
DP. <i>See</i> Dynamic programming.			
Dual heuristic programming (DHP)	61		
Dual-spool turbofan engine			
high-bypass	202f	203	
uses	201		
Dynamic programming (DP)	279	319	323
direct neural	34		
formulation	60		
Dynamic uncertainty. <i>See</i> Process			
uncertainty.			

<u>Index Terms</u>	<u>Links</u>	
E		
EGT sensor. <i>See</i> Exhaust gas temperature sensor.		
EHSV. <i>See</i> Electrohydraulic servo valve.		
EKF. <i>See</i> Extended Kalman filter.		
Electrohydraulic servo valve (EHSV)	246–247	
Elitist selection	240	
EMU. <i>See</i> Engine monitoring unit.		
Engine health		
degradation	223	
monitoring	216	
Engine monitoring unit (EMU)	220–221	
Engine performance deterioration		
mitigating control (EPDMC)	213	214f
FADEC	214	
normalized thrust	215f	216
Engine pressure ratio (EPR)	205	220
Engineering data	174	
EOM. <i>See</i> Equations of motion.		
EPDMC. <i>See</i> Engine performance deterioration mitigating control.		
Episode	276	
EPR. <i>See</i> Engine pressure ratio.		
Equations of motion (EOM)	7	
aircraft	21	
in body-fixed coordinate system	36	
for inverted pendulum model	289	
nonholonomic unicycle robot	341	

Index Terms

Links

Exhaust gas temperature sensor (EGT
sensor)

205

Extended Kalman filter (EKF)

314

357

F

F-16 short-period dynamics flight

control

desired control

79

optimality conditions

78

problem description

76–78

result analysis

angle of attack histories

78f

constrained control and desired

control

82f

desired and actual pitch rate

histories

79f

true and estimated uncertainties

80f

81f

weights histories

83f

system dynamics

77

uncertainty estimation

79

84

FAA. *See* Federal Aviation

Administration.

FADEC. *See* Full authority digital engine

control.

Failure modes and effects analysis

(FMEA)

185

Failure to seat

197

Fault detection

210

222

Fault isolation

222

FCL. *See* Functional capability level.

This page has been reformatted by Knovel to provide easier navigation.

Index Terms

Links

Federal Aviation Administration			
(FAA)	212		
Feed-forward network architecture	210		
Feedback control systems	60		
adaptive-critic-based controller	61		
control inputs	62		
network types	60		
NN uses	61		
nonquadratic cost function	63		
SNAC	61		
steady-state HJB theory	62		
Feedback controller-based information-			
state roadmap (FIRM)	311–312	346	347
breaking curse of history	349		
characteristic	352		
efficient planning	349		
FIRM MDP	351		
framework	350		
generic construction	352		
incorporating obstacles in planning	349		
inducing reachable belief nodes	349		
overall policy	351		
in POMDP	321		
probabilistic completeness	349–350		
absorption probability of local			
planners	362		
extended stopping region	361	362	
hyperstate	361		
Liouville–Neumann series	362–364	374–381	

<u>Index Terms</u>	<u>Links</u>		
Feedback controller-based information-state roadmap (FIRM) (<i>Cont.</i>)			
local planner stopping region	361	362	
notation	361		
steps in construction	350		
Field of view (FOV)	447		
FIPA. <i>See</i> Foundation for Intelligent Physical Agent.			
FIRM. <i>See</i> Feedback controller-based information-state roadmap.			
Flapping flights	1		
<i>See also</i> RoboBat.			
control possibilities and concerns	3		
wing motions	4f		
FLC. <i>See</i> Fuzzy logic controller.			
Flight control law, L_1 adaptive	142		
<i>See also</i> L_1 adaptive control in flight.			
evaluation	148		
flight under stability degradation	148		
pitch-doublet response	149f	151f	153f
	155f		
roll-doublet response	150f	152f	154f
L_1 control in large envelope modeling tasks			
departure-prone edge exploration	165	166	167
	168		
research tasks supported by L_1			
adaptive control law	158t		
unsteady aerodynamic modeling			
work	165		

Index Terms

Links

Flight control law, L_1 adaptive (*Cont.*)

offset-to-landing tasks 155

comments and CHR 157t

performance criteria 156

requirements 156

system with unmatched nonlinear

uncertainty 143

adaptive architecture theorem 146–147

adaptive state-feedback controller 145

control law 146

state predictor 146

unmatched component 144

tuning 147–148

Flight mechanics 14–15

Floquet theory 2

FMEA. *See* Failure modes and effects

analysis.

FMU. *See* Fuel metering unit.

Foundation for Intelligent Physical Agent

(FIPA) 394 400

FOV. *See* Field of view.

Fuel metering unit (FMU) 246

Full authority digital engine control

(FADEC) 201

engine control design 204

acceleration schedules 206 207

burst-chop 207 208f

challenges in 204–205

constraint 205

control gains 209

Index Terms

Links

Full authority digital engine control		
(FADEC) (<i>Cont.</i>)		
deceleration schedules	206	207
engine limit protection	206	
lead-lag compensator	206	
modern engine control logic	205	
performance map	208	
turbofan engine	202	
components	203f	
dynamic engine models	204	
operational limitations	203–204	
station designations	203f	
Fully actuated point robot	337	
<i>See also</i> Nonholonomic unicycle robot.		
GPRM and GRRT	338	339
GPRM comparison with PRM	339f	341f
GRRT comparison with RRT	340f	342f
local feedback controllers	338	
PRM and RRT	339	340
robot motion model	337–338	
Functional capability level (FCL)	175	184
Fuzzifier	232	
<i>See also</i> Defuzzifier.		
measurement	232	
membership function values	233	
nonzero membership	232	
Fuzzy logic controller (FLC)	248	
defuzzifier	234	
fuzzifier	232	
measurement	232	

<u>Index Terms</u>	<u>Links</u>		
Fuzzy logic controller (FLC) (<i>Cont.</i>)			
membership function values	233		
nonzero membership	232		
inference engine	234–238		
benefits	236		
drawbacks	236	238	
FLC components	234		
fuel system test bench	238		
nonisosceles fuzzy membership			
functions	237f		
AND operator	236		
OR operator	236		
single-antecedent rules	236		
water heater control rule base	235t		
water heater output fuzzy sets	235f	236	
rule base	233		
cruise control system	233	234t	
IF-THEN rules	233–234		
structure	233f		
Fuzzy membership functions	249	259f	270
FLC rule base	251t		
nonisosceles	237f	249	250f
	252f		
parameters	251t	258	
surface plot	260	261	
symmetric	250f		

G

Gas-turbine fuel system

Index Terms

Links

Gas-turbine fuel system (*Cont.*)

See also Genetic fuzzy gas-turbine fuel
system.

actuation system	246	
EHSV model	246–247	
sensing system model	247	248
closed-loop fuel control system	247f	
FLC	248	
gain scheduling technique	248	
hardware models	246	
turbine engines	246	
Gaseous nitrogen (GN)	197	
Gaseous oxygen (GOX)	197	
Gaussian belief space	353	
Generalized probabilistic roadmaps		
(GPRM)	311	332–333
algorithm	333–335	
fully actuated point robot	338	
generic-construction	329	
generic-planning on	330	
in MDP	320	
performance on unicycle robot	343f	344f
robot motion under	334f	
Generalized rapidly exploring random		
tree (GRRT)	330	336
algorithm	336–337	
comparison with RRT	340f	342f
fully actuated point robot	338	
performance on unicycle robot	343f	344f

Index Terms

Links

Generalized sampling-based motion		
planners	332	
<i>See also</i> Sampling-based feedback		
motion planners (SFMP).		
fully actuated point robot	337	
GPRM and GRRT	338	339
GPRM comparison with PRM	339f	341f
GRRT comparison with RRT	340f	342f
local feedback controllers	338	
PRM and RRT	339	340
robot motion model	337–338	
GPRM	332–333	
algorithm	333–335	
robot motion under	334f	
GRRT	336–337	
model	332	
nonholonomic unicycle robot	341	342
DOF system	344–345	
GRRT and GPRM performance	343f–344f	
uncertainty	342–343	
Generic transport model aircraft (GTM		
aircraft)	117	141
<i>See also</i> Modified reference model		
MRAC (M-MRAC).		
AirSTAR	148	155
L_1 control law design	147	
with stick-to-surface control	156	
uncertain model	120	

Index Terms

Links

Genetic algorithms	230	238	
benefits	261–262		
binary mutation	241	242f	
crossovers	240		
one-point	240	241f	
two-point	241f		
elitist selection	240		
fixed-length binary-coded strings	239		
fuzzy-neural system optimization	244		
genetic inversion	241	242f	243f
performance comparison	256t	257	
population members	238	239	
population selection	242	243f	
Genetic fuzzy gas-turbine fuel			
system	248		
controller design	249		
closed-loop simulation	254		
closed-loop system performance	253		
crossovers	255		
genetic algorithm population	254		
genetic variation	255		
input and output	249		
mutations	255		
parameters	252		
rise time	254		
steady-state error	253–254		
controller performance analysis	255		
characteristic parameters	258t		
fitness function value	257f		
FLC step responses	261f	262	

<u>Index Terms</u>	<u>Links</u>		
Genetic fuzzy gas-turbine fuel			
system (<i>Cont.</i>)			
fuzzy controller	260f		
fuzzy membership functions	259f		
fuzzy system performance			
chart	257t		
genetic algorithm performance			
comparison	256t		
performance measure	258f		
FLC comparison	270–271		
fuzzy membership functions	249	250f	
FLC rule base	251t		
nonisosceles	249	250f	252f
parameters	251t		
Genetic fuzzy systems	229–230	248	
FLC	230f	233f	
flexibility	230		
fuzzy set	230		
genetic algorithms	238–244		
membership functions	230		
for set temperature	232f		
for temperature	231f		
stochastic robustness analysis	244–246		
Glenn Research Center (GRC)	176	211	
Gliding flight	1	3	5
	29		
GN. <i>See</i> Gaseous nitrogen.			
Goman and Khrabrov’s model	9–10		
GOX. <i>See</i> Gaseous oxygen.			

Index Terms

Links

GPRM. *See* Generalized probabilistic roadmaps.

GRC. *See* Glenn Research Center.

GRRT. *See* Generalized rapidly exploring random tree.

GTM aircraft. *See* Generic transport model aircraft.

H

HADS. <i>See</i> Health Assessment Database System.			
Hamilton–Jacobi–Bellman equation			
(HJB equation)	35	42–43	60
constrained optimal control algorithm	62–63		
Lipschitz continuous	65		
optimality condition	65–66		
for SOLA	44		
system dynamics	65		
Hardware (HW)	179		
HASI. <i>See</i> Human–autonomous system interaction.			
HDP. <i>See</i> Heuristic dynamic programming.			
Health and Usage Monitoring System			
(HUMS)	177		
Health Assessment Database System			
(HADS)	193		
Health parameters	205	218–219	221
Health-enabled system control	191		

<u>Index Terms</u>	<u>Links</u>		
Helicopter dynamic model	37f		
backstepping technique	38		
challenges	39		
kinematics	37		
mass-inertia matrix	38		
nonlinear aerodynamic effects	38		
rotational transformation matrix	37		
translational rotation matrix	37		
travel direction	36		
Heuristic dynamic programming (HDP)	60	61	
Hierarchical reinforcement learning			
(HRL)	273	289	313
High-pressure compressor (HPC)	203	204	208
High-pressure turbine (HPT)	203	204	
High-reliability engine control (HREC)	209	210	
AANN	210f	211	
dual hardware redundancy architecture	209		
low-speed sensor failure	211f		
HJB equation. <i>See</i> Hamilton–Jacobi– Bellman equation.			
HKF. <i>See</i> Hybrid Kalman filter.			
HMI. <i>See</i> Human–machine–interface.			
Hopf oscillator	5		
HPC. <i>See</i> High-pressure compressor.			
HPT. <i>See</i> High-pressure turbine.			
HREC. <i>See</i> High-reliability engine control.			
HRL. <i>See</i> Hierarchical reinforcement learning.			
Human–autonomous system interaction			
(HASI)	404		

<u>Index Terms</u>	<u>Links</u>		
Human-machine-interface (HMI)	449	453	454
<i>See also</i> Unmanned aerial vehicle(UAV).			
model of formation dynamics	455		
tracking control law	456		
user-observable subspace	455		
user-predictable subspace	455		
vehicle fleet centroid	455		
HUMS. <i>See</i> Health and Usage Monitoring System.			
HW. <i>See</i> Hardware.			
Hybrid Kalman filter (HKF)	223		
Hyperstate	361		

I

IE. <i>See</i> Information exposure.			
IEEE. <i>See</i> Institute of Electrical and Electronics Engineers.			
IEEE 1451 family of standards	179f		
IEEE 1451. 1	179		
IEEE 1451. 3	179–180		
IEEE 1451. 4	180		
IEEE P1451. 0	178	179	
IEEE P1451. 5	180		
IEEE P1451. 6	180		
P1451. 3, P1451. 4	178		
for SS&A	178		
wireless communication protocol			
standards	180		
iISHM. <i>See</i> Intelligent integrated systems health management.			

Index Terms

Links

iISHM domain model (iISHM-DM)	181	189	193f
abstract representation	183f		
development	184		
software environment development	186		
software system requirements	184–185		
DIaKA	182	183f	
distribution within and across net-			
works	185		
leak detection in isolated subsystems	196f		
object class definitions for	195f		
root-cause tree instantiation	197f		
sensor processes	183–184		
and system image	192f		
iISHM implementation standards	177	180–181	
IEEE 1451 family of standards	178–180		
OSA-CBM standard	180		
iISHM-DM. <i>See</i> iISHM domain model.			
ILEC. <i>See</i> Intelligent life extending			
control.			
Inference engine	185	234	
AND operator	236		
drawbacks	236	238	
FLC components	234		
fuel system test bench	238		
KB	194		
nonisosceles fuzzy membership			
functions	237f		
OR operator	236		
single-antecedent rules	236		

Index Terms

Links

Inference engine (*Cont.*)

water heater control rule base 235t

water heater output fuzzy sets 235f 236

Information 174

Information exposure (IE) 397

MAS 439

requirements 398t

Information-state. *See* Belief.

Institute of Electrical and Electronics

Engineers (IEEE) 178

INT. *See* Interaction.

Integrated Resilient Aircraft Control

(IRAC) 140

Integrated Systems Health Management

(ISHM) 174 184

Intelligent integrated systems health

management (iISHM) 174

See also iISHM domain model

(iISHM-DM).

capability development 175

iISHM implementation standards 177–181

intelligent sensors and components 186–187

layered iISHM capability 176f

NASA GRC 176

PITEX 177

sensor selection and placement 187f–189f

software capabilities 184–186

integration of subsystems 189–190

intelligent control 191

SI&E 189

Index Terms

Links

Intelligent integrated systems health		
management (iISHM) (<i>Cont.</i>)		
in system design	190	
verification and validation		
advancement	192	
Intelligent life extending control (ILEC)	211–212	
Intelligent sensor (IS)	174	186
functionality	192	
implementation	187f	
Interaction (INT)	397	
Inverted pendulum system	288f	289
action-value function	291	292
agent task	291	
inverted pendulum casting	289	
constants	290t	
motion equations	290	
Q-learning parameters	291t	
Q-learning reward structure	290t	
Q-learning simulation conditions	292t	
reinforcement learning controller	289	
using fuzzy controller	288	289
IRAC. <i>See</i> Integrated Resilient Aircraft		
Control.		
IS. <i>See</i> Intelligent sensor.		
ISHM. <i>See</i> Integrated Systems Health		
Management.		
ISHM domain model (ISHM-DM)	193	194f
ISHM-DM. <i>See</i> ISHM domain model.		
Isolated subsystems (IsoSubs)	195	197
IsoSubs. <i>See</i> Isolated subsystems.		

Index Terms

Links

J

J-SNAC controller	
actual plant	67
dynamic reoptimization	67
neural network structure	69
optimal weight differences	69
tracking problem	69–70
virtual plant	67
J-SNAC synthesis. See	
Cost-function-based single	
network adaptive critic	
synthesis.	

K

Kalman filter	218		
KB. <i>See</i> Knowledge base.			
Kennedy Space Center (KSC)	180	181	182
Kinematic state machine (KSM)	407		
Knowledge	174		
Knowledge base (KB)	194		
KSC. <i>See</i> Kennedy Space Center.			
KSM. <i>See</i> Kinematic state machine.			

L

L ₁ adaptive control for NPS autonomous	
UAV	132
inner-loop L ₁ augmentation loop	133f

<u>Index Terms</u>	<u>Links</u>		
L ₁ adaptive control for NPS autonomous UAV (<i>Cont.</i>)			
L ₁ adaptation for path-following missions			
aircraft configuration	136		
benefits	138		
closed-loop system	136f		
flight-test results	136	137	138
path-following performance	137f		
L ₁ adaptation in control surface			
failures	138		
ten-deg locked-in-place left-aileron			
failure	140		
two-deg locked-in-place left-aileron			
failure	139f		
L ₁ adaptive output-feedback			
controller	132		
adaptation law	134		
adaptive architecture theorem	135–136		
closed-loop behavior	134		
control signal	135		
output predictor	134		
single-input plant	132–133		
single-output plant	132–133		
L ₁ adaptive control in flight			
conventional adaptive control			
systems	129		
fast adaptation	131		
feature	131		
limitations	129–130		

Index Terms

Links

L ₁ adaptive control in flight (<i>Cont.</i>)		
loss-of-control accident data	130f	
performance requirements	130	
safety standpoint	130	
Launch Complex 20 (LC-20)	180–181	182f
LC-20. <i>See</i> Launch Complex	20	
Learner	276	
Least-mean-square (LMS)	452	
Life extending control	211	
FAA requirements	212	
ILEC	211–212	
optimized acceleration schedules		
for takeoff acceleration process	212f	
thrust response curves for	213f	
Life-cycle counts	216	
Limit-cycle oscillator. <i>See</i> Hopf oscillator.		
Linear quadratic Gaussian (LQG)	316	
controllers	353	
stationary	353–355	
time-varying	353	354
Linear quadratic Gaussian motion-		
planning (LQG-MP)	316	
Linear quadratic regulator (LQR)	61	289
Linear variable displacement transducer		
(LVDT)	247	
Linear-time-invariant (LTI)	117	
Liouville–Neumann series	362–364	374–381
Liquid oxygen (LOX)	193	
LMS. <i>See</i> Least-mean-square.		

Index Terms

Links

Local controllers	322	324	356
in FIRM	347		
macro-actions vs.	325		
roadmap of	324–325		
Local planner stopping region	361	362	
Low-pressure compressor (LPC)	203	204	208
Low-pressure turbine (LPT)	203	204	
LOX. <i>See</i> Liquid oxygen.			
LPC. <i>See</i> Low-pressure compressor.			
LPT. <i>See</i> Low-pressure turbine.			
LQG. <i>See</i> Linear quadratic Gaussian.			
LQG-based FIRM construction	352		
FIRM nodes	355–356		
Gaussian belief space	353		
local controller	356		
LQG controllers	353		
off-line construction	357–358		
online phase algorithm	358–359		
planning with	358		
stationary LQG	353–355		
time-varying LQG	353	354	
underlying PRM	355		
LQG-MP. <i>See</i> Linear quadratic Gaussian motion-planning.			
LQR. <i>See</i> Linear quadratic regulator.			
LTI. <i>See</i> Linear-time-invariant.			
Lunar scientific sampling mission	388		
investigation	439		
safety concerns	389–390		

<u>Index Terms</u>	<u>Links</u>		
LVDT. <i>See</i> Linear variable displacement transducer.			
Lyapunov analysis	44		
M			
M-MRAC. <i>See</i> Modified reference model MRAC.			
Machine Information Management Open Standards Alliance (MIMOSA)	178	180	
Manager structure	408		
cognitive manager	409		
conversation manager	409		
decision manager	410		
robot agent's cognitive capability replacement	409f		
status dimensions	409	410f	
MARHES. <i>See</i> Multi-Agent Robotics Hybrid and Embedded Systems.			
MARHES testbed	467f		
Markov decision process (MDP)	277	310	319
<i>See also</i> Partially observed Markov decision process (POMDP).			
characterization	319		
DP equation	319–320		
FIRM	347	350	351
	357		
GPRM	320		
induced MDP by SFMP	325–327		
robotic motion-planning problem	330		
Markovian property	310		

Index Terms

Links

MAS. <i>See</i> Multi-agent systems.		
Mass-inertia matrix	38	
MAV. <i>See</i> Micro aerial vehicle.		
Maximum likelihood (ML)	317	
MAXQ	274	
MAXQ-Q	274	
MDP. <i>See</i> Markov decision process.		
Micro aerial vehicle (MAV)	1	
animal flight	1	
analysis by Floquet theory	2	
regimes	1	3
robotic flapping flyers	2	
artificial flapping fliers	2	
biological fliers	2	
control law design	16–20	
flight mechanics with articulated		
wings	14	
dihedral deflections	15	
wing dihedral use	14f	
yaw control effectiveness	15	
limitation	25	
perching	3	
PID controllers	16	
properties	24t	
robotic bat	2f	
MIMO. <i>See</i> Multi-input multi-output.		
MIMOSA. <i>See</i> Machine Information		
Management Open Standards		
Alliance.		

Index Terms

Links

Mission Society	390		
<i>See also</i> Multi-agent-based subsystems and astronaut interactions.			
actors	390–391		
behavior norms	391–392		
impact on framework design	392		
participants in	393t		
primary values	391		
protocol	394		
protocols	400		
software agent paradigm	394–395		
stakeholders in	390		
ML. <i>See</i> Maximum likelihood.			
MMHT. <i>See</i> Multiple model hypothesis testing.			
Model predictive control approach (MPC approach)	224	225f	
Model tuning parameters	219		
Model-based control and diagnostics	216	217f	218
<i>See also</i> Full authority digital engine control (FADEC).			
adaptive control	224		
actuators	226		
model predictive control implementation	225f		
MPC approach	224		
MPC-MIMO controller improvements	225f		
SISO controller	226		
EPR	220		
Kalman filter	218		

<u>Index Terms</u>	<u>Links</u>		
Model-based control and diagnostics (<i>Cont.</i>)			
model tuning parameters	219		
onboard condition monitoring	220		
EMU	220	221	
engine health degradation	223		
fault detection	222		
fault isolation	222		
HKF approach	223		
information flow for turbine engine gas	221f		
MMHT	222	223	
off-line and online engine gas path			
diagnostics	224f		
performance diagnostics	221		
SNR	222		
unfaulted engine	222		
thrust estimation accuracy comparison	219f		
thrust response comparison	220f		
tuning parameter vector	219		
Modified reference model MRAC (M-MRAC)	92f	93	
aerospace application	117		
asymptotic properties	100–101		
disturbance rejection properties	108–111		
input tracking performance	122f	124f	125
nominal dynamics characteristics			
closed-loop	120t		
open-loop	117	120t	
output tracking performance	121f	123f	125
preliminaries	93–97		
reference model modification			
MIMO uncertain linear system	98–100		

Index Terms

Links

Modified reference model MRAC (M-MRAC) (*Cont.*)

scalar example	111		
step response	112f	113f	
in external disturbance	119f		
for large adaptation rates	114f		
system's response			
for different initial conditions	118f		
to sinusoidal command	115f		
to square wave command	116f		
transient properties	101–108		
Monte Carlo analysis	245–246		
Monte Carlo methods	279		
Monte Carlo simulation	262	287–288	303
closed-loop system	262		
discretization levels	305		
fitness function values for	265f	266f	
multiple thermals in two-dimensions	304f	305f	
performance parameter values			
histogram	268f		
step input results comparison	267f		
using episodes	305–306		
Motion planning of robots	312		
<i>See also</i> Decision making under uncertainty.			
deterministic approaches	312–312		
PRM and RRT techniques	313		
process uncertainty	314–315		
SMDP	313–314		
stochastic/uncertain maps	314		
uncertainty in robotic systems	313		

Index Terms

Links

Motion planning problem	310		
<i>See also</i> Decision making under uncertainty.			
FIRM	311–312		
motion uncertainty	311		
optimal substructure	312		
PRM method	310	311	
RRT	310	311	
sensing uncertainty	311		
Motion uncertainty. <i>See</i> Process uncertainty.			
MPC approach. <i>See</i> Model predictive control approach.			
MRAC	92		
<i>See also</i> Modified reference model MRAC (M-MRAC).			
step response	112f	113f	
system's response			
to sinusoidal command	115f		
to square wave command	116f		
Multi-Agent Robotics Hybrid and Embedded Systems (MARHES)	463		
Multi-agent systems (MAS)	387	388	392
<i>See also</i> Multi-agent-based subsystems and astronaut interactions.			
additional requirements on	399t		
information exposure requirements	398t		
protocol development	399	400	

Index Terms

Links

Multi-agent systems (MAS) (*Cont.*)

protocol utilization

autonomy mode change protocol	406–407	
dependencies among MAS	404	406f
HASI framework	404	
nonvirtual MAS information	404	405f
Power Distribution Device Access		
Virtual Team	405	

Multi-agent-based subsystems and

astronaut interactions

See also Mission Society.

autonomous systems for sensor

information assessment	395
------------------------	-----

behavior requirements	397
-----------------------	-----

coordinator role	397
------------------	-----

individual use case scenarios	397t
-------------------------------	------

information exposure requirements	398t
-----------------------------------	------

information on subsystem functioning	395
--------------------------------------	-----

motivating scenarios	396
----------------------	-----

procedures	396
------------	-----

role constructs	397	398
-----------------	-----	-----

Multi-input multi-output (MIMO)	98	218	459
---------------------------------	----	-----	-----

Multiple model hypothesis testing

(MMHT)	222	223f
--------	-----	------

Multiresolution state-space discretization

method	274	283
--------	-----	-----

multiple goal regions	285–286
-----------------------	---------

for N dimensions	283
--------------------	-----

coarse grid	283f
-------------	------

finer grid	284f
------------	------

This page has been reformatted by Knovel to provide easier navigation.

Index Terms

Links

Multiresolution state-space discretization	
method (<i>Cont.</i>)	
learning problem	285
multiresolution learning	284
using N -dimensional case	284
state-space discretization	283
policy comparison	286–287
direct	287
performance-based	287–288
purpose	275
reinforcement learning	276
two-and N -dimensional continuous	
domain	281–283
2D discrete case	282
2D state space	281f
applications	282–283
continuous state space	282
continuous-domain learning	
problem	281
curse of dimensionality	281
optimal action-value function	281

N

NASA AirSTAR flight-test vehicle, L_1	
adaptive control for	143f
flight control law evaluation	148–157
GTM aircraft	141
L_1 adaptive flight control law	142–148
L_1 flight control law	141
in large envelope modeling tasks	158–167

Index Terms

Links

NASA Kennedy Space Center (NASA KSC)	181	
NASA KSC. <i>See</i> NASA Kennedy Space Center.		
NASA SSC. <i>See</i> NASA Stennis Space Center.		
NASA Stennis Space Center (NASA SSC)	181	
National Institute of Standards and Technology (NIST)	178	
NATO. <i>See</i> North Atlantic Treaty Organization.		
Naval Postgraduate School (NPS)	132	
NCAP. <i>See</i> Network Capable Application Processor.		
Network Capable Application Processor (NCAP)	179	191
Neural network (NN)	34	
basis function	47	
for constrained optimal control problems	62	
solving HJB equations	63	
training	66–67	68f
uncertainty	69f	73
NIST. <i>See</i> National Institute of Standards and Technology.		
NN. <i>See</i> Neural network.		
Nonholonomic unicycle robot	341	342
<i>See also</i> Fully actuated point robot.		
DOF system	344–345	
GRRT and GPRM performance	343f–344f	
uncertainty	342–343	

<u>Index Terms</u>	<u>Links</u>		
Nonlinear discrete-time affine systems	35		
North Atlantic Treaty Organization (NATO)	216		
NPS. <i>See</i> Naval Postgraduate School.			
O			
OBEM. <i>See</i> Onboard engine model.			
Occupancy-grid (OG)	338		
ODE. <i>See</i> Ordinary differential equation.			
OG. <i>See</i> Occupancy-grid.			
OLA. <i>See</i> Online approximator.			
Onboard engine model (OBEM)	216	217	223
Online approximator (OLA)	35	43–44	
dynamic controller	35		
parameter estimation error	45–46		
reconstruction error	44		
tuning algorithm	45		
Online phase algorithm	358–359		
Open Systems Architecture for			
Condition-Based Maintenance			
standard (OSA-CBM standard)	180		
Open Systems Architecture for Enterprise			
Application Integration standard			
(OSA-EAI standard)	180		
Ordinary differential equation (ODE)	10		
OSA-CBM standard. <i>See</i> Open Systems			
Architecture for Condition-			
Based Maintenance standard.			
OSA-EAI standard. <i>See</i> Open Systems			
Architecture for Enterprise			
Application Integration standard.			

<u>Index Terms</u>	<u>Links</u>		
Output-feedback controller, L_1 adaptive	132		
adaptation law	134		
adaptive architecture theorem	135–136		
closed-loop behavior	134		
control signal	135		
output predictor	134		
single-input plant	132–133		
single-output plant	132–133		
P			
Partially observed Markov decision			
process (POMDP)	310	320	
<i>See also</i> Markov decision process			
(MDP).			
belief MDP problem	320		
FIRM	321		
FIRM solution	352		
local controllers	322		
Particle RRT algorithm (pRRT			
algorithm)	314		
Path isolation-based analysis	317		
Path-following missions, L_1			
adaptation for			
aircraft configuration	136		
benefits	138		
closed-loop system	136f		
flight-test results	136	137	138
path-following performance	137f		
pdf. <i>See</i> probability density function.			

Index Terms

Links

Pennsylvania State University's Applied Research Laboratory (PSU-ARL)	181	
Percent overshoot (%OS)	253	
%OS. <i>See</i> Percent overshoot.		
Percent steady-state error (%SSE)	253	
%SSE. <i>See</i> Percent steady-state error.		
Perching	3	21
aircraft and data acquisition	24	
torque limitations.	25	
VICON motion-capture system	24–25	
aircraft motion	21	
angle-of-attack control	25f	26
distance from landing point	23	
flight path control	26–27	
initial flight path angle	23	
initial flight speed	21–22	
lateral-directional control	26	
maneuver	14	21
perching maneuver demonstration	27–28	
Personal Satellite Assistant (PSA)	388	
Phase synchronization	6	29
PI controller. <i>See</i> Proportional-integral controller.		
PID. <i>See</i> Proportional-integral-derivative.		
Piecewise linear Kalman filter (PLKF)	223	
Pitching. <i>See</i> Twisting.		
PITEX. <i>See</i> Propulsion IVHM Technology Experiment.		
PLA. <i>See</i> Power lever angle.		

Index Terms

Links

PLKF. *See* Piecewise linear Kalman filter.

Plunging. *See* Flapping.

POD. *See* Probability-of-detection.

POMDP. *See* Partially observed Markov
decision process.

Potential field methods 312

Power Distribution Device Access Virtual
Team 405

Power lever angle (PLA) 204 207

PR. *See* Pressure ratio.

Pressure ratio (PR) 208

Prioritized target search (PTS) 449 460

PRM. *See* Probabilistic roadmaps.

Probabilistic completeness

in deterministic case 359

difference with deterministic case 360

globally successful policy 360

initial belief 360

path isolation-based analysis 317

space-covering-based analysis 317–318

successful policy 360

under uncertainty 359 360–361

Probabilistic roadmaps (PRM) 310

edge dependence problem 348

to FIRM 346

generalization 311 312

medial-axis-based 314

in motion-planning problems 313

underlying 324 355

Probability density function (pdf) 312

<u>Index Terms</u>	<u>Links</u>		
Probability-of-detection (POD)	447		
Problem formulation, UAV			
framework structure and implementation	448f	449	
hybrid automata of hierarchical			
structure	449f		
problem description	447		
POD map	447		
problem	448		
PTS algorithm	449		
quadrotor UAV fleet	447f		
Process and sensing uncertainty			
computed off-line	316		
FIRM	315–316		
collision probability	317		
feedback policy	316–317		
LQG-MP method	316		
online replanning	316		
probabilistic completeness			
path isolation-based analysis	317		
space-covering-based analysis	317–318		
Process uncertainty	309	311	313
	318		
in dynamical models	314–315		
motion-planning problem	310		
SFMP with	330–331		
Proportional-integral controller (PI controller)	237	248	
Proportional-integral-derivative			
(PID)	11		
Propulsion IVHM Technology Experiment (PITEX)	177		

Index Terms

Links

Protocol	394		
FIPA	394		
performative	394		
proximity alert	422	423	
proximity avoid	422		
proximity safing	419	421–422	
Protocol development	387	390	413
<i>See also</i> Multi-agent-based subsystems and astronaut interactions.			
astronaut initiated message exchange	415t		
autonomy mode change protocol	400		
coordinator/lead agent role	400	401	
for nonvirtual MAS	400	401f	
for virtual MAS	402–403		
complex protocol	414	415	
for future space exploration missions	387		
iterative process features	414t		
robot-to-robot team messaging activity	415t		
safety	400		
TWE protocol	413		
for use in MAS	399		
Protocol utilization in MAS			
autonomy mode change protocol	406–407		
dependencies among MAS	404	406f	
HASI framework	404		
nonvirtual MAS information	404	405f	
Power Distribution Device Access Virtual Team	405		

<u>Index Terms</u>	<u>Links</u>		
Proximity			
alert protocol	422	423	
avoid protocol	422		
safing protocol	419	421–422	
pRRT algorithm. <i>See</i> particle RRT algorithm.			
PSA. <i>See</i> Personal Satellite Assistant.			
Pseudocontrol hedging method	34		
Pseudogoals	285		
PSU-ARL. <i>See</i> Pennsylvania State University’s Applied Research Laboratory.			
PTS. <i>See</i> Prioritized target search.			
Q			
Q-learning algorithm	280–281	292	
<i>See also</i> Inverted pendulum system.			
with AAG	294		
cart velocity	298		
inverted pendulum AAG goal region			
equations	296t		
inverted pendulum action sets	295t		
inverted pendulum controller	295f	296f	297f
state quantizations	295t		
time history and phase diagram	297		
using simulations	294		
using value function	295		
controller	289	293f	294f
hierarchical	274		
jagged nature	293		

<u>Index Terms</u>	<u>Links</u>		
Q-learning algorithm (<i>Cont.</i>)			
policy color scheme	293t		
policy representation	292		
stopping criteria	286–287		
time history and phase diagram	293	294	
value function	292		
Quadrotor baseline controller design			
hardware experiments	463	464	465
MARHES testbed	467f		
nonlinear quadrotor model			
velocities	464f		
quadrotor velocity profile	465f		
testbed and communication			
links	466f		
Vicon motion capture system	464	465	
waypoint tracking comparison	463f		
Wi-Fi routers	464		
ZigBee	464		
hybrid framework simulations	460		
behavior changes	460	462	
initial probability of detection			
map	461f		
reduced probability of detection			
map	461f		
tracking algorithm	462f		
low-level controller design			
cascade PD controller	460f		
feedback linearization	458–459		
one-to-one mapping	459		
proposed controller	460		

<u>Index Terms</u>	<u>Links</u>		
Quadrotor baseline controller design (<i>Cont.</i>)			
quadrotor modeling			
aircraft dynamics	457		
hummingbird quadrotor	457f		
kinematics	457		
nonlinear quadrotor model	458		
notation symbols	458t		
quadrotor dynamics	456		
Quadrotor modeling			
aircraft dynamics	457		
hummingbird quadrotor	457f		
kinematics	457		
nonlinear quadrotor model	458		
notation symbols	458t		
quadrotor dynamics	456		
Quanser pendulum platform	7		
R			
Rapid Flight Test Prototyping System			
(RFTPS)	132		
Rapidly exploring random tree (RRT)	310	330	
comparison with GRRT	340f	342f	
extension	314		
in motion-planning problems	313		
Recharging protocol	419		
phases	419		
thread of message exchanges	420		
Reinforcement learning (RL)	276	280	310
Bellman equation	278		
DP and TD learning	279		

Index Terms

Links

Reinforcement learning (RL) (*Cont.*)

episode	276
MDPs	277
Monte Carlo methods	279
RL problem	280–281

Research and Technology Organization

(RTO)	216
-------	-----

Retrofit intelligent engine control 209

See also Full authority digital engine control (FADEC).

EPDMC	213	214f
FADEC	214	
normalized thrust	215f	216
high-reliability engine control	209	
AANN	210f	211
dual hardware redundancy		
architecture	209	
low-speed sensor failure	211f	
life extending control	211	
FAA requirements	212	
ILEC	211	
optimized acceleration schedules	213f	
takeoff acceleration process	212f	

RFTPS. *See* Rapid Flight Test Prototyping System.

Rise time (Trise) 253

RL. *See* Reinforcement learning.

Roadmap technique 312

RoboBat

See also Central pattern generator (CPG).

Index Terms

Links

RoboBat (*Cont.*)

flight mode change results	13f	14	
PID controllers	11		
pitch control results	11f	12f	13
pitch stability results	12f		
testbed	2f		
velocity control	14		
wing motions	4f		
RoboBat lead-lag motion	4		
Robot agent	407		
<i>See also</i> Autonomous robot teams.			
activities	407		
control laws	407–408		
Cybele™ agent	407		
within DCF structure	408f		
decision-making structures	410		
allowable values of state			
dimensions	410t		
decision modules	411f		
protocol-related foci	411		
development	408		
manager structure	408		
cognitive manager	409		
conversation manager	409		
decision manager	410		
robot agent's cognitive capability			
replacement	409f		
status dimensions	409	410f	

Index Terms

Links

Robot-initiated safety protocol		
proximity alert protocol	422	423
proximity avoid protocol	422	
proximity safing protocol	419	421–422
Robotic bat	2f	
Robustness analysis	262	
<i>See also</i> Genetic fuzzy gas-turbine fuel system.		
controller robustness assessment	264	
EHSV	263	
LVDT	263	
Monte Carlo simulation	262	
fitness function values	265f	
fitness function values histogram	266f	
performance parameter values histogram	268f	
step input results comparison	267f	
stochastic robustness analysis performance	269	
Rocket-engine test facility	192	
<i>See also</i> Intelligent integrated systems health management (iISHM).		
core pilot iISHM	192	
CSG LOX system	193f	
iISHM-DM and image of system	192f	
iISHM-DM	193	
object class definitions for	195f	
IsoSubs	195	
functional diagram	196f	
root-cause tree for	196f	197f

Index Terms

Links

Rocket-engine test facility (<i>Cont.</i>)		
knowledge-based iISHM-DM	194	
Rotational transformation matrix	37	
RRT. <i>See</i> Rapidly exploring random tree.		
RTO. <i>See</i> Research and Technology Organization.		
 S		
S&As. <i>See</i> Sensors and actuators.		
S4. <i>See</i> Systematic Sensor Selection Strategy.		
Sampling-based feedback motion planners (SFMP)	322	
<i>See also</i> Generalized sampling-based motion planners; Motion planning problem.		
collision detection module	331	
decision making under uncertainty	345–346	
edge dependence problem	346	348f
FIRM	346	347
breaking curse of history	349	
characteristic	352	
computed costs for node-controller pairs	371t	
efficient planning	349	
FIRM MDP	351	
framework	350	
generic construction	352	
incorporating obstacles in planning	349	

Index Terms

Links

Sampling-based feedback motion			
planners (SFMP) (<i>Cont.</i>)			
inducing reachable belief nodes	349		
overall policy	351		
planning	352	370	372f
	373		
probabilistic completeness	349–350	361–364	
replanning	370	372f	373
steps in construction	350		
generic algorithms			
generic-construction of generalized			
roadmap	329		
generic-planning on generalized			
roadmap	330		
hierarchical planning in levels	331f		
incorporating obstacles	327–328		
larger environment	373–374		
LQG-based FIRM construction	352–353		
FIRM nodes	355–356	366	
Gaussian belief space	353		
local controller	356		
LQG controllers	353		
off-line construction	357–358		
online phase algorithm	358–359		
planning with	358		
stationary LQG	353–355		
time-varying LQG	353	354	
underlying PRM	355		
motion model	365		
observation model	365–366		

Index Terms

Links

Sampling-based feedback motion		
planners (SFMP) (<i>Cont.</i>)		
obstacle-free construction		
dynamic programming	323	
induced MDP by SFMP	325–327	
local controllers	324–325	
overall policy	327	
planning problem	324	
probabilistic completeness	325	
SFMP nodes	324	
underlying PRM for SFMP	324	
preliminaries		
feedback controllers	322	
feedback policy	323	
landing probability	323	
reachability	323	
stopping region	322–323	
probabilistic completeness		
deterministic and probabilistic case		
difference	360	
in deterministic case	359	
feasible initial belief	360	
globally successful policy	360	
successful policy	360	
under uncertainty	359	360–361
with process uncertainty and stochastic		
maps	330	
success probability	328–329	
transition probabilities and costs	356–357	366–370
SAS. <i>See</i> Stability augmentation system.		

Index Terms

Links

Semi Markov decision process (SMDP)	310		
hierarchical	311		
issues	313		
modified DP formulation	325		
Sensing system model	247	248	
Sensing uncertainty	301	309	313
	318		
in dynamical models	315–317		
POMDP	320		
sampling-based motion planner	314		
Sensor icons	183		
Sensors and actuators (S&As)	178		
Serial peripheral interface (SPI)	179		
Settling time (Tset)	253		
SFMP. <i>See</i> Sampling-based feedback motion planners.			
SI&E. <i>See</i> Systems integration and engineering.			
Signal-to-noise ratio (SNR)	222		
Simulated autonomous robot team effort, protocol utilization in			
Active Conversations console	423		
astronaut's exclusion radius	424		
decision logic	428		
problem report	426		
recharge procedure	426	427f	
simulation results	426		
worker robot activities	423	424f	425f
	427	428f	

Index Terms

Links

Single network adaptive critic controller (SNAC controller)	48	61
Single online approximator (SOLA)	35	
adaptive approach	35	
optimal control of helicopter	43–46	
Single-input single-output (SISO)	218	
SISO. <i>See</i> Single-input single-output.		
SM. <i>See</i> Stall margins.		
Smart sensor (SS)	186	
Smart sensors and actuators (SS&As)	178	
SMDP. <i>See</i> Semi Markov decision process.		
SNAC controller. <i>See</i> Single network adaptive critic controller.		
SNR. <i>See</i> Signal-to-noise ratio.		
Software agent paradigm	394–395	
SOLA. <i>See</i> Single online approximator.		
Space exploration missions		
protocol development for future	387	
robot's role	388	
Space shuttle main engine (SSME)	176–177	
Space-covering-based analysis	317–318	
Spacecraft control problem	70	
nonquadratic cost function	72	
optimality conditions	71	
problem description	70–71	
result analysis		
constrained control histories	75f	
Euler angle time histories	73f	
spacecraft angular rate histories	74f	

Index Terms

Links

Spacecraft control problem (<i>Cont.</i>)	
uncertainty histories	75f
uncertainty estimation	73–74
SPI. <i>See</i> Serial peripheral interface.	
SS. <i>See</i> Smart sensor.	
SS&As. <i>See</i> Smart sensors and actuators.	
SSC. <i>See</i> Stennis Space Center.	
SSME. <i>See</i> Space shuttle main engine.	
Stability augmentation system (SAS)	142
Stakeholders	390
Stall margins (SM)	208
Steady-state HJB theory	62
Stennis Space Center (SSC)	181
Stochastic motion roadmap	315
Stochastic robustness analysis	246f
controller performance	245
controller robustness	244
Monte Carlo analysis	245–246
Styrofoam cubes	428
Sweeping. <i>See</i> RoboBat lead-lag motion.	
Swing-up maneuver	289
Systematic Sensor Selection Strategy (S4)	187–189
Systems integration and engineering (SI&E)	189

T

Takagi–Sugeno model	288
TCC. <i>See</i> Test Control Center.	
TCP/IP. <i>See</i> Transmission control protocol/Internet protocol.	

Index Terms

Links

TD learning. *See* Temporal-difference learning.

Team leader robot	418–419	435
activities	412–413	
decision modules in	411f	
status dimensions for	410t	
TWE protocol	416–417	
Teamwork effort protocol (TWE protocol)	413	416
astronaut oversight	417	418t
finite state machine view	416f	
sequences	416	417
simpler protocols	417t	
team leader robot	418–419	
worker robot agents	418	

TEDS. *See* Transducer electronic data sheet.

Temporal-difference learning (TD learning) 279

Test Control Center (TCC) 192

Thermal location casting

initial updraft field limits	301t	
reinforcement learning agent	299	
as reinforcement learning problem	298	299
sink region	301	
thermal location axis definitions	300t	
updraft field	299	300
vertical velocity	300	301

Three-wheel omnidirectional mobile robot 365

Throttle resolver angle (TRA) 207

TIM. *See* Transducer interface module.

TRA. *See* Throttle resolver angle.

Tracking filter 218

Index Terms

Links

Transducer electronic data sheet (TEDS)	178	180	192
Transducer interface module (TIM)	179		
Translational rotation matrix	37		
Transmission control protocol/Internet protocol (TCP/IP)	407		
Trise. <i>See</i> Rise time.			
Tset. <i>See</i> Settling time.			
Tuning parameter vector	219		
Turbofan engine	202	203	
AANN approach	210	211	
components	203f		
dual-spool	201		
dynamic engine models	204		
modern	209		
multistage compressors	203		
turbines	203		
VBV and VSV	203		
TWE protocol. <i>See</i> Teamwork effort protocol.			
Twisting	4f		

U

UAV. <i>See</i> Uninhabited air vehicle; Unmanned aerial vehicle.			
UGV. <i>See</i> Unicycle unmanned ground vehicle.			
Uncertainty in environment map	313		
Uncertainty roadmap	314		
Unicycle unmanned ground vehicle (UGV)	448		
Uninhabited air vehicle (UAV)	275		

Index Terms

Links

University of New Mexico (UNM)	463	
UNM. <i>See</i> University of New Mexico.		
Unmanned aerial vehicle (UAV)	33	
<i>See also</i> Adaptive sampling-based filter for target tracking (AFTT); L_1 adaptive control for NPS autonomous UAV.		
advantages	445	
Bayesian filtering techniques	446	
CSAT mission management algorithm	446	
fleet of quadrotor UAVs	447f	
prioritized target search control algorithm	450	
coordination algorithm	449–450	
prioritized search algorithm	450f	
Unmanned helicopter		
backstepping-based controller	34	
controlling problem	34	
difficulties	33	
discrete-time equation	35	
HJB equation	42–43	
kinematic controller	39	
landing position and orientation	55f	
NN control scheme control inputs	47	
HJB tracking problem	47	
state tracking error	46	
NN-based controllers	34	35

Index Terms

Links

Unmanned helicopter (*Cont.*)

nonlinear discrete-time affine		
systems	35	
OLA-based dynamic controller	35	
optimal tracking control scheme	39f	
optimality and convergence	48	
position vs. tracking time	54f	
pseudocontrol hedging method	34	
simulation results	52	
regulating ability	53f	
takeoff and circular maneuver		
position	54f	
tracking	52	
SOLA based optimal control	43	
Lyapunov analysis	44	
OLA parameter estimation error	45–46	
OLA reconstruction error	44	
OLA representation	43–44	
OLA tuning algorithm	45	
SOLA-based adaptive approach	35	
solutions	33–34	
system stability	48–52	
tracking control	34	
underactuated helicopter system	34	
virtual controller	36	40
angular velocity and orientation		
relationship	41–42	
yaw control	41	

<u>Index Terms</u>	<u>Links</u>		
V			
Variable bleed valve (VBV)	203	208	
Variable stator vanes (VSV)	203	208	
VBV. <i>See</i> Variable bleed valve.			
Vicon motion capture system	24	25	464
	465		
Virtual controller	36	40–42	
Virtual Intelligent Sensor Environment			
(VISE)	187		
intelligent sensor implementation	187		
software architecture	193		
transformation	192		
VISE. <i>See</i> Virtual Intelligent Sensor			
Environment.			
Visualization and reporting (VR)	397		
VR. <i>See</i> Visualization and reporting.			
VSV. <i>See</i> Variable stator vanes.			
W			
Wi-Fi routers	464		
Wireless communication protocol standards	180		
Worker robot agents	418		
interaction	418		
TWE protocol	416		
values of selected state dimensions for	410t		
Z			
ZE. <i>See</i> Zero fuzzy set.			

Index Terms

Links

Zero fuzzy set (ZE)	249
ZigBee routers	464